EVOLMPNN: PREDICTING MUTATIONAL EFFECT ON HOMOLOGOUS PROTEINS BY EVOLUTION ENCODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting protein properties is paramount for biological and medical advancements. Current protein engineering mutates on a typical protein, called the wild-type, to construct a family of homologous proteins and study their properties. Yet, existing methods easily neglect subtle mutations, failing to capture the effect on the protein properties. To this end, we propose EVOLMPNN, Evolution-aware Message Passing Neural Network, to learn evolution-aware protein embeddings. EVOLMPNN samples sets of anchor proteins, computes evolutionary information by means of residues and employs a differentiable evolution-aware aggregation scheme over these sampled anchors. This way EVOLMPNN can capture the mutation effect on proteins with respect to the anchor proteins. Afterwards, the aggregated evolution-aware embeddings are integrated with sequence embeddings to generate final comprehensive protein embeddings. Our model shows up to 6.4% better than state-of-the-art methods and attains $36 \times$ inference speedup in comparison with large pre-trained models. The code and models are available at https://anonymous.4open.science/r/EvolMPNN.

1 INTRODUCTION

Can we predict important properties of a protein by directly observing only the effect of a few mutations on such properties? This basic biological question (Wells, 1990; Fowler & Fields, 2014) has recently engaged the machine learning community due to the current availability of benchmark data (Rao et al., 2019; Dallago et al., 2021; Xu et al., 2022). Proteins are sequences of amino-acids (residues), which are the cornerstone of life and influence a number of metabolic processes, including diseases (Pauling et al., 1951; Ideker & Sharan, 2008). For this reason, protein engineering stands at the forefront of modern biotechnology, offering a remarkable toolkit to manipulate and optimise existing proteins for a wide range of applications, from drug development to personalised therapy (Ulmer, 1983; Carter, 2011; Alley et al., 2019).

One fundamental process in protein engineering progressively mutates an initial protein, called the wild-type, to study the effect on the protein’s properties (Siezen et al., 1991). These mutations form a family of homologous proteins as in Figure 1. This process is appealing due to its cheaper cost compared to other methods and reduced time and risk (Wang et al., 2012; Engqvist & Rabe, 2019).

Yet, the way mutations affect the protein’s properties is not completely understood (Bryant et al., 2021; Sarkisyan et al., 2016; Wu et al., 2016), as it depends on a number of chemical reactions and bonds among residues. For this reason, machine learning offers a viable alternative to model complex interactions among residues. Initial approaches employed feature engineering to capture protein’s evolution (Saravanan & Gautham, 2015; Feng & Zhang, 2000); yet, a manual approach is expensive and does not offer enough versatility. Advances in NLP and CV inspired the design of deep protein sequence encoders (Hochreiter & Schmidhuber, 1997; Yu et al., 2017; Vaswani et al., 2017) and general purpose Protein Language Models (PLMs) that are pre-trained on large scale datasets of sequences. Notable PLMs include ProtBert (Brandes et al., 2022), AlphaFold (Jumper et al., 2021), TAPE Transformer (Rao et al., 2019) and ESM (Rives et al., 2021). These models mainly rely on Multiple Sequence Alignments (MSAs) (Meier et al., 2021) to search on large databases of protein evolution. Nevertheless, this search process is insensitive to subtle yet crucial mutations and introduces additional computational burdens (Pearson, 2013; Chatzou et al., 2016).
To overcome the limitations of previous models, we propose EVOLMPNN, Evolution-aware Message Passing Neural Network, to predict the mutational effect on homologous proteins. Our fundamental assumption is that there are inherent correlations between protein properties and the sequence differences among them, as shown in Figure 1(b). EVOLMPNN integrates both protein sequence and evolutionary information by identifying where and which mutations occur on the target protein sequence, compared with known protein sequences and predicts the mutational effect on the target protein property. To avoid the costly quadratic pairwise comparison among proteins, we devise a theoretically grounded (see Section 4.6) linear sampling strategy to compute differences only among the proteins and a fixed number of anchor proteins (Section 4.2). We additionally introduce two extensions of our model, EVOLGNN and EVOLFORMER, to include available data on the relation among proteins (Section 4.5). The theoretical computation complexity of proposed methods are provided to guarantee their efficiency and practicality. We apply the proposed methods to three benchmark homologous protein property prediction datasets with nine splits. Empirical evaluation results (Section 5.1) show up to 6.7% Spearman’s $\rho$ correlation improvement over the best performing baseline models, reducing the inference time by $36\times$ compared with pre-trained PLMs.

Figure 1: Protein property prediction on homologous protein family. (a) An example homologous protein family with labelled nearby mutants with few mutations. The task is to predict the label of unknown mutants with more mutations. (b) The evolutionary pattern for (a); $(p_1, m_1)$ indicates the mutation $m_1$ at the $p_1$-th position of the protein sequence.

## 2 Preliminary and Problem

In the protein engineering process, we first receive a set of proteins $\mathcal{M} = \{P_i\}_{i=1,2,\ldots,M}$ in which each protein can be associated with a label vector $Y_i \in \mathbb{R}^n$ that describes its biomedical properties, e.g., fitness, stability, fluorescence, solubility, etc. Each protein $P_i$ is a linear chain of amino-acids $P_i = \{r_j\}_{j=1,2,\ldots,N}$. While a protein sequence folds into specific 3D conformation to perform some biomedical functions, each amino-acid is considered as a residue. Residues are connected to one another by peptide bonds and can interact with each other by different chemical bounds (Pauling et al., 1951). In short, the function of a protein is mainly determined by the chemical interactions between residues. Since the 3D structure is missing in benchmark datasets (Rao et al., 2019 Dallago et al., 2021 Xu et al., 2022), we assume no 3D protein information in this paper.

**Homologous Protein Family.** A set of protein sequences $\mathcal{M}$ is a homologous protein family if there exists an ancestral protein $P_{WT}$, called wild-type, such that any $P_i \in \mathcal{M}$ is obtained by mutating $P_{WT}$ through substitution, deletion, insertion and truncation of residues (Ochoterena et al., 2019). As shown in Figure 1(a), a homologous protein family can be organised together by representing their evolutionary relationships and Figure 1(b) illustrates the detailed evolutionary patterns.

**Research Problem.** Protein engineering based on homologous proteins is a promising and essential direction for designing novel proteins of desired properties (Huang et al., 2014; Langan et al., 2019). Understanding the relation between protein sequence and property is one essential step. Practically, biologists perform experiments in the lab to label the property $Y_{\text{TRAIN}}$ of a set of protein $\mathcal{M}_{\text{TRAIN}} \subset \mathcal{M}$ and the follow-up task is predicting $Y_{\text{TEST}}$ of the rest proteins $\mathcal{M}_{\text{TEST}} \subset \mathcal{M}$. However, homologous proteins typically have similarities in their amino-acid sequences, structures, and functions due to their shared ancestry. Accurately predicting the homologous protein property by distinguishing these subtle yet crucial differences is still an open challenge.
3 RELATED WORK

**Feature Engineering.** Besides conducting manual experiments in labs to measure protein properties, the basic solution is to design different feature engineering methods based on relevant biological knowledge, to extract useful information from protein sequence (Klein et al., 1985; Feng & Zhang, 2000; Wang et al., 2017). Dallago et al. (2021) introduce using Levenshtein distance (Li & Liu, 2007) and BLOSUM62-score (Eddy, 2004) relative to wild-type to design protein sequence features. In another benchmark work, Xu et al. (2022) adopt another two typical protein sequence feature descriptors, i.e., Dipeptide Deviation from Expected Mean (DDE) (Saravanan & Gautham, 2015) and Moran correlation (Moran) (Feng & Zhang, 2000). For more engineering methods, refer to the comprehensive review (Lee et al., 2007).

**Protein Representation Learning.** In the last decades, empowered by the outstanding achievements of machine learning and deep learning, protein representation learning has revolutionised protein property prediction research. Early work along this line adopts the idea of word2vec (Mikolov et al., 2013) to protein sequences (Xu et al., 2018; Mejía-Guerra & Buckler, 2019). To increase model capacity, deeper protein sequence encoders were proposed by the Computer Vision (CV) and Nature Language Processing (NLP) communities (Hochreiter & Schmidhuber, 1997; Yu et al., 2017; Vaswani et al., 2017). The latest works develop Protein Language Models (PLMs), which focus on employing deep sequence encoder models for protein sequences and are pre-trained on million- or billion-scale sequences. Well-known works include ProtBert (Brandes et al., 2022), AlphaFold (Jumper et al., 2021), TAPE Transformer (Rao et al., 2019) and ESM (Rives et al., 2021). However, most existing work does not pay enough attention to these subtle yet crucial differences in homologous proteins. Rives et al. (2021); Jumper et al. (2021) explore protein Multiple Sequence Alignments (MSAs) (Rao et al., 2021a; Meier et al., 2021) to capture the mutational effect. Nevertheless, the MSA searching process introduces additional computational burden and is insensitive to subtle but crucial sequence differences (Pearson, 2013). Chatzou et al. (2016) indicate the shortcomings of MSAs on easily neglecting the presence of minor mutations, which can propagate errors to downstream protein sequence representation learning tasks. This paper also lies in this direction, we propose a novel solution for the challenging homologous protein property prediction tasks.

4 FRAMEWORK

![Figure 2: Our EVOLMPNN framework encodes protein mutations via a sapient combination of residue evolution and sequence encoding.](image-url)

EVOLMPNN is a novel framework that integrates both protein sequence information and evolution information by means of residues. As a result, EVOLMPNN accurately predicts the mutational effect on homologous protein families. First, in Section 4.1, we introduce embedding initialisation for protein sequence and residues and the update module for residue embedding (Section 4.2). The evolution encoding in Section 4.3 is the cornerstone of the model that ameliorates protein embeddings.
We conclude in Section 4.4 with the generation of final proteins embeddings and model optimisation. We complement our model with a theoretical analysis to motivate our methodology and a discussion of the computation complexity (Section 4.6). We additionally propose extended versions of EVOLMPNN that deal with available protein-protein interactions (Section 4.5).

4.1 Embedding Initialisation

**Protein Sequence Embedding.** Given a set of proteins \( \mathcal{M} = \{ P_i \}_{i=1,2,\ldots,M} \), we first adopt a (parameter-frozen) PLM model \(^1\) (Rao et al., 2021b; Meier et al., 2021) as protein sequence encoder to initialise the protein sequence embedding \( (H) \) for every protein \( P_i \), which include macro (i.e., protein sequence) level information as the primary embedding.

\[
H = \text{PLM} \text{ENCODER}(\{ P_i \}_{i=1,2,\ldots,M}),
\]

where the obtained protein embedding \( H \in \mathbb{R}^{M \times d} \) and \( H_i \) corresponds to each protein \( P_i \). Different encoders can extract information on various aspects, however, existing PLM models that rely on MSAs are not sensitive enough to capture the evolution pattern information in homologous protein families (Pearson, 2013; Chatzou et al., 2016) systematically indicate the shortcomings of MSAs on easily neglecting the presence of minor mutations, which can propagate errors to downstream protein sequence representation learning tasks.

**Residue Embedding.** In order to properly capture the evolution information in homologous proteins, we delve into the residue level for micro clues. We adopt two residue embedding initialisation approaches, i.e., one-hot encoding \( (\Phi^{OH}) \) and pre-trained PLM encoder \( (\Phi^{PLM}) \), to generate protein’s initial residue embeddings \( \tilde{X}_i = \{ x_j^i \}_{j=1,2,\ldots,N} \), where \( x_j^i \in \mathbb{R}^d \). In particular, \( \Phi^{OH} \) assigns each protein residue \( j \) with a binary feature vector \( x_j^i \), where \( x_j^i = 1 \) indicates the appearance of the \( b \)-th residue at \( P_i \)’s \( j \)-th position. By stacking \( N \) residues’ feature vectors into a matrix, we can obtain \( X_i \in \mathbb{R}^{N \times d} \). On the other hand, following the benchmark implementations (Zhu et al., 2022), \text{PLM} \text{ENCODER} can export residue embeddings similar to Eq. 1. Formally, \( \Phi^{PLM} \) initialises protein residue embeddings as \( X_i = \text{PLM} \text{ENCODER}(\{ r_j \}_{j=1,2,\ldots,N}) \).

**Position Embedding.** Another essential component of existing PLM is the positional encoding, which was first proposed by Vaswani et al. (2017). This positional encoding effectively captures the relative structural information between entities and integrates it with the model (Ying et al., 2021). In our case, correctly recording the position of each residue in the protein sequence plays an essential role in identifying each protein’s corresponding mutations. Because the same mutation that occurs at different positions may lead to disparate influences on protein property. Therefore, after initialising residue embeddings, we further apply positional embedding on each protein’s residues. We adopt a methodology that reminisces (Raoss et al., 2023) that demonstrates the paramount importance of assigning each residue with a unique position embedding. As such, we randomly initialise a set of \( d \) position embeddings \( \Phi^{Pos} \in \mathbb{R}^{N \times d} \). We denote the residue embedding empowered by position embedding as \( X_i = X_i \odot \Phi^{Pos} \).

4.2 Residue Embedding Update

The 3D protein folding depends on the strength of different chemical bonds between residues to maintain a stable 3D structure. Previous studies carefully designed residue contact maps to model the residue-residue interactions to learn effective residue embeddings (Rao et al., 2021b; Gao et al., 2023). In this paper, we adopt the residue-residue interaction to update residue embeddings but eschew the requirement of manually designing the contact map. Instead, we assume the existence of an implicit fully connected residue contact map of each protein \( P_i \) and implement the Transformer model (Vaswani et al., 2017; Wu et al., 2022) to adaptively update residue embeddings. Denote \( R^{(\ell)}_i \) as the input to the \((\ell + 1)\)-th layer, with the first \( R^{(0)}_i = \tilde{X}_i \) be the input encoding. The \((\ell + 1)\)-th

---

1We do not fine-tune PLM in this paper for efficiency consideration.
2There are 20 different amino-acid residues commonly found in proteins.
layer of residue embedding update module can be formally defined as follows:

$$\text{Att}^h_i(R^{(l)}_i) = \text{SOFTMAX} \left( \frac{R^{(l)}_i W^{f,h}_Q (R^{(l)}_i W^{f,h}_K)^T}{\sqrt{d}} \right),$$

$$\hat{R}^{(l)}_i = R^{(l)}_i + \sum_{h=1}^{H} \text{Att}^h_i(R^{(l)}_i) R^{(l)}_i W^{f,h}_V W^{f,h}_O,$$

(2)

where $W^{f,h}_O \in \mathbb{R}^{d_H \times d}$, $W^{f,h}_Q$, $W^{f,h}_K$, $W^{f,h}_V \in \mathbb{R}^{d \times d_H}$, $W^{f}_1 \in \mathbb{R}^{d \times r}$, $W^{f}_2 \in \mathbb{R}^{d \times d}$, $H$ is the number of attention heads, $d_H$ is the dimension of each head, $d_1$ is the dimension of the hidden layer, ELU [Clevert et al., 2015] is an activation function, and $\text{Att}^h_i(R^{(l)}_i)$ refers to the attention matrix. After each Transformer layer, we add a normalisation layer i.e., LayerNorm [Ba et al., 2016], to reduce the over-fitting problem proposed by Vaswani et al. (2017). After stacking $L_r$ layers, we obtain the final residue embeddings as $R_i = R^{(L_r)}_i$.

4.3 Evolution Encoding

In homologous protein families, all proteins are mutants derived from a common wild-type protein $P_{WT}$ with different numbers and types of mutations. In this paper, we propose to capture the evolutionary information via the following assumption.

**Assumption 1** (Protein Property Relevance). Assume there is a homologous protein family $M$ and a function $\text{DIFF} \text{IFF}$ can accurately distinguish the mutations on mutant $P_i$ compared with any $P_j$ as $\text{DIFF} \text{DIFF}(P_i, P_j)$. For any target protein $P_i$, its property $Y_i$ can be predicted by considering 1) its sequence information $P_i$; 2) $\text{DIFF}(P_i, P_j)$, and the property of $P_j$, i.e., $Y_j$. Shortly, we assume there exists a function $f$ that maps $Y_i \leftarrow f(\text{DIFF}(P_i, P_j), Y_j)$.

Motivated by Assumption [1], we take both protein sequence and the mutants difference $\text{DIFF}(P_i, P_j)$ to accurately predict the protein property. To encode the protein sequence, we employ established tools described in Section [4.1]. Here instead, we describe the evolution encoding to realise the function of $\text{DIFF}(P_i, P_j)$.

The naïve solution to extract evolutionary patterns in a homologous family is constructing a complete phylogenetic tree [Fitch & Margoliash, 1967] based on the mutation distance between each protein pair. Yet, finding the most parsimonious phylogenetic tree is NP-hard (Sankoff, 1975).

To address the aforementioned problems, we propose an anchor-based protein evolution encoding method. Specifically, denote $H_i^{(l)}$ as the input to the $(l + 1)$-th block and define $H_i^{(0)} = H_i$. The evolution localisation encoding of the $(l + 1)$-th layer contains the following key components: (i) $k$ anchor protein $\{P_{S_j}\}_{j=1,2,...,k}$ selection. (ii) Evolutionary information encoding function $\text{DIFF}$ that computes the difference between residues of each protein and those of the anchor protein, and target protein’s evolutionary information is generated by summarising the obtained differences:

$$d_{ij} = \text{COMBINE}(R_i - R_{S_j}),$$

(3)

where COMBINE can be implemented as differentiable operators, such as, CONCATENATE, MAX POOL, MEAN POOL and SUM POOL; here we use the MEAN POOL to obtain $d_{ij} \in \mathbb{R}^d$. (iii) Message computation function $F_{\text{MESSAGE}}$ that combines protein sequence feature information of two proteins with their evolutionary differences. We empirically find that the simple element-wise product between $H_j^{(l)}$ and $d_{ij}$ attains good results:

$$F_{\text{MESSAGE}}(i, j, H_j^{(l)}, d_{ij}) = H_j^{(l)} \odot d_{ij},$$

(4)

(iv) Aggregate messages from $k$ anchors and combine them with protein’s embedding as follow:

$$\hat{H}_i^{(l)} = \text{COMBINE}(\{F_{\text{MESSAGE}}(i, j, H_j^{(l)}, d_{ij})\}_{j=1,2,...,k}),$$

(5)

$$H_i^{(l+1)} = \text{CONCAT}(H_i^{(l)}, \hat{H}_i^{(l)}) W^l,$$

(6)

where $W^l \in \mathbb{R}^{2d \times d}$ transform concatenated vectors to the hidden dimension. After stacking $L_p$ layers, we obtain the final protein sequence embedding $Z^p_i = H_i^{(L_p)}$. 

5
4.4 Final Embedding and Optimisation

After obtaining protein $P_i$’s residue embeddings $R_i$ and sequence embedding $Z_i^p$, we summarise its residual embeddings as a vector $Z_i^p = \text{MEAN\ POOL}(R_i)$. The final protein embedding summarises the protein sequence information and evolution information as the comprehensive embedding $Z_i = \text{CONCAT}(Z_i^p, Z_i^e)$ and the final prediction is computed as $Y_i = Z_iW^\text{FINAL}$ where $W^\text{FINAL} \in \mathbb{R}^{d \times \theta}$, $\theta$ is the number of properties to predict. Afterwards, we adopt a simple and common strategy, similar to (Xu et al., 2022), to solve the protein property prediction tasks. Specifically, we adopt the MSE loss $L$ to measure the correctness of model predictions on training samples against ground truth labels. The objective of learning the target task is to optimise model parameters to minimise the loss $L$ on this task. The framework of EVOLMPNN is summarised in Algorithm 1 in Appendix A.

4.5 Extensions on Observed Graph

EVOLMPNN does not leverage any information from explicit geometry among proteins, where each protein only communicates with randomly sampled anchors (Section 4.3). However, it is often possible to have useful structured data $G = (\mathcal{M}, A)$ that represents the relation between protein-protein by incorporating specific domain knowledge (Zhong et al., 2023). Therefore, here we introduce EVOLGNN, an extension of EVOLMPNN on the possibly observed protein interactions.

**EVOLGNN.** We compute the evolution information as Eq. (3). The evolution information can be easily integrated into the pipeline of message-passing neural networks, as an additional structural coefficient $W_{ij}^{\text{Wijesinghe & Wang}, 2022}$:

$$m_i^{(t)} = \text{AGGREGATE}^N(\{A_{ij}, d_{ij} \} | j \in N(i)), \quad \text{Evol. info.}$$

$$m_i^{(t)} = \text{AGGREGATE}^T(\{A_{ij}, d_{ij} \} | j \in N(i)) \cdot H_i^{(t-1)}, \quad \text{Evol. info.}$$

$$H_i^{(t)} = \text{COMBINE}(m_i^{(t)}, m_i^{(t)}), \quad \text{Evol. info.}$$

where AGGREGATE$^N(\cdot)$ and AGGREGATE$^T(\cdot)$ are two parameterised functions. $m_i^{(t)}$ is a message aggregated from its neighbours $N(i)$ of protein $P_i$, and their structure $A_{ij}$ and evolution $d_{ij}$ coefficients. $m_i^{(t)}$ is an updated message from protein $P_i$ after performing an element-wise multiplication between AGGREGATE$^T(\cdot)$ and $H_i^{(t-1)}$ to account for structural and evolution effects from its neighbours. After, $m_i^{(t)}$ and $m_i^{(t)}$ are combined together to obtain the update embedding $H_i^{(t)}$.

**EVOLFORMER.** Another extension relies on pure Transformer structure, which means the evolution information of $\mathcal{M}$ can be captured by every protein. The evolution information can be integrated into the pipeline of Transformer, as additional information to compute the attention matrix:

$$\text{Att}^E(H^{(t)}) = \text{SOFTMAX}\left(\frac{H^{(t)}W_{Q}^{(t)}(H^{(t)}W_{K}^{(t)})^T}{\sqrt{d}} + \text{MEAN\ POOL}(\{R_i\}_{i=1,2,\ldots,M})\right), \quad \text{Evol. info.}$$

Other follow-up information aggregation and feature vector update operations are the same as the basic Transformer pipeline, as described in Eq. (2).

4.6 Theoretical Analysis

**Anchor Selection.** Inspired by (You et al., 2019), we adopt Bourgain’s Theorem (Bourgain, 1985) to guide the random anchor number $k$ of the evolution encoding layer. Briefly, support by a constructive proof (Theorem 2, Linial et al., 1995) of Bourgain Theorem (Theorem 1), only $k = O(\log^2 M)$ anchors are needed to ensure the resulting embeddings are guaranteed to have low distortion (Definition 1), in a given metric space $(\mathcal{M}, F_{\text{Dist}})$. EVOLMPNN can be viewed as a generalisation of the embedding method of Theorem 2 where $F_{\text{Dist}}(\cdot)$ is generalised via message passing functions.

---

Available contact map describes residue-residue interactions can be easily integrated as relational bias of Transformer (Wu et al., 2022) as we used in Section 4.2.
Therefore, Theorem 2 offers a theoretical guide that \(O(\log^2 M)\) anchors are needed to guarantee low distortion embedding. Following this principle, EVOLMPNN choose \(k = \log^2 M\) random anchors, denoted as \(\{S_j\}_{j=1,2,\ldots,\log^2 M}\), and we sample each protein in \(M\) independently with probability \(\frac{1}{2^j}\). Detailed discussion and proof refer to Appendix B.

**Complexity Analysis.** The computation costs of EVOLMPNN, EVOLGNN, and EVOLFORMER come from residue encoding and evolution encoding, since the protein sequence and residue feature initialisation have no trainable parameters. The residue encoder introduces the complexity of \(O(MN)\) following an efficient implementation of NodeFormer (Wu et al., 2022). In the evolution encoding, EVOLMPNN performs communication between each protein and \(\log^2 M\) anchors, which introduces the complexity of \(O(M \log^2 M)\); EVOLGNN performs communication between each protein and \(K\) neighbours with \(O(KM)\) complexity; EVOLFORMER performs communication between all protein pairs, which introduces the complexity of \(O(M)\), following the efficient implement, NodeFormer. In the end, we obtain the total computation complexity of EVOLMPNN - \(O((N + \log^2 M)M)\), EVOLGNN - \(O((N + K)M)\) and EVOLFORMER - \(O((N + 1)M)\).

## 5 EXPERIMENTAL STUDY

In this section, we empirically study the performance of EVOLMPNN. We validate our model on three benchmark homologous protein family datasets and evaluate the methods on nine data splits to consider comprehensive practical use cases. Our experiments comprise a comprehensive set of state-of-the-art methods from different categories. We additionally demonstrate the effectiveness of two extensions of our model, EVOLGNN and EVOLFORMER, with different input features. We conclude our analysis studying the influence of some hyper-parameters and investigating the performance of EVOLMPNN on high mutational mutants.

### Datasets and Splits.** We perform experiments on benchmark datasets of several important protein engineering tasks, including AAV, GB1, and Fluorescence, and generate three splits on each data set. Data statistics are summarised in Table 1. The split \(\lambda\text{-VS-REST}\) indicates that we train models on wild-type protein and mutants of no more than \(\lambda\) mutations, while the rest are assigned to test. The split \(\text{LOW-VS-HIGH}\) indicates that we train models on sequences with target value scores equal to or below wild-type, while the rest are assigned to test. For more details refer to Appendix C.

**Baselines.** As baseline models, we consider methods in four categories. First, we selected four feature engineer methods, i.e., Levenshtein (Dallago et al., 2021), BLOSUM62 (Dallago et al., 2021), DDE (Saravanan & Gautham, 2015) and Moran (Feng & Zhang, 2000). Second, we select four protein sequence encoder models, i.e., LSTM (Hochreiter & Schmidhuber, 1997), Transformer (Rao et al., 2019), CNN (Rao et al., 2019) and ResNet (Yu et al., 2017). Third, we select four pre-trained PLM models, i.e., ProtBert (Elmagarr et al., 2022), ESM-1b (Rives et al., 2021), ESM-1v (Meier et al., 2021) and ESM-2 (Lin et al., 2023). In the end, we select four GNN-based methods which can utilise available graph structure, i.e., GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), GraphTransformer (Shi et al., 2021) and NodeFormer (Wu et al., 2022).

**Implementation.** We follow the PEER benchmark setting including train and test pipeline, model optimisation and evaluation method (evaluation is Spearman’s \(p\) metric), adopted in (Xu et al., 2022) to make sure the comparison fairness. For the baselines, including feature engineer, protein sequence

---

Table 1: Datasets splits, and corresponding statistics; if the split comes from a benchmark paper, we report the corresponding citation.

<table>
<thead>
<tr>
<th>Landscape</th>
<th>Split</th>
<th># Total</th>
<th>#Train</th>
<th>#Valid</th>
<th>#Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAV (Bryant et al., 2021)</td>
<td>2-VS-REST (Dallago et al., 2021)</td>
<td>82,583</td>
<td>28,626</td>
<td>3,181</td>
<td>50,776</td>
</tr>
<tr>
<td></td>
<td>7-VS-REST (Dallago et al., 2021)</td>
<td>82,583</td>
<td>63,001</td>
<td>7,001</td>
<td>12,581</td>
</tr>
<tr>
<td></td>
<td>LOW-VS-HIGH (Dallago et al., 2021)</td>
<td>82,583</td>
<td>42,791</td>
<td>4,755</td>
<td>35,037</td>
</tr>
<tr>
<td>GB1 (Wu et al., 2016)</td>
<td>2-VS-REST (Dallago et al., 2021)</td>
<td>8,733</td>
<td>381</td>
<td>43</td>
<td>8,309</td>
</tr>
<tr>
<td></td>
<td>3-VS-REST (Dallago et al., 2021)</td>
<td>8,733</td>
<td>2,691</td>
<td>299</td>
<td>5,743</td>
</tr>
<tr>
<td></td>
<td>LOW-VS-HIGH (Dallago et al., 2021)</td>
<td>8,733</td>
<td>4,580</td>
<td>509</td>
<td>3,644</td>
</tr>
<tr>
<td>Fluorescence (Sarkisyan et al., 2016)</td>
<td>2-VS-REST</td>
<td>54,025</td>
<td>12,712</td>
<td>1,413</td>
<td>39,900</td>
</tr>
<tr>
<td></td>
<td>3-VS-REST (Xu et al., 2022)</td>
<td>54,025</td>
<td>21,446</td>
<td>5,362</td>
<td>27,217</td>
</tr>
<tr>
<td></td>
<td>LOW-VS-HIGH</td>
<td>54,025</td>
<td>44,082</td>
<td>4,899</td>
<td>5,044</td>
</tr>
</tbody>
</table>

[https://github.com/DeepGraphLearning/PEER_Benchmark]
Effective homologous protein property prediction. Notably, GNN-based methods that utilise manually con-
duction is not an effective solution.

We perform additional experiments on GB1 datasets to investigate the performance of two extended models, i.e., EVOLGNN and EVOLFORMER and study the influence of different residue embedding initialisation methods. The results summarised in Table 3 evince that EVOLMPNN outperforms the other two variants in three splits, and all our proposed models outperform the best baseline. This result confirms the effectiveness of encoding evolution information for homologous protein property prediction. Besides, the models adopting the PLM encoding $\phi_{PLM}$ achieve better performance than

<table>
<thead>
<tr>
<th>Category</th>
<th>Model</th>
<th>2-vs-R.</th>
<th>4-vs-R.</th>
<th>L-vs-H.</th>
<th>3-vs-R.</th>
<th>L-vs-H.</th>
<th>Fluorescence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Engineer</td>
<td>Levenshtein</td>
<td>0.575</td>
<td>0.550</td>
<td>0.251</td>
<td>0.156</td>
<td>-0.069</td>
<td>-0.108</td>
</tr>
<tr>
<td></td>
<td>BLOSUM62</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>0.128</td>
<td>0.005</td>
<td>-0.127</td>
</tr>
<tr>
<td></td>
<td>DDE</td>
<td>0.649†</td>
<td>0.636</td>
<td>0.158</td>
<td>0.445†</td>
<td>0.816</td>
<td>0.306</td>
</tr>
<tr>
<td></td>
<td>Moran</td>
<td>0.337†</td>
<td>0.398</td>
<td>0.069</td>
<td>0.069†</td>
<td>0.389</td>
<td>0.193</td>
</tr>
<tr>
<td>Protein Seq. Encoder</td>
<td>LSTM</td>
<td>0.125†</td>
<td>0.608</td>
<td>0.308</td>
<td>-0.002</td>
<td>-0.002</td>
<td>-0.007</td>
</tr>
<tr>
<td></td>
<td>Transformer</td>
<td>0.681†</td>
<td>0.748</td>
<td>0.304</td>
<td>0.271†</td>
<td>0.877</td>
<td>0.474</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
<td>0.746†</td>
<td>0.730</td>
<td>0.406</td>
<td>0.502†</td>
<td>0.857</td>
<td>0.515</td>
</tr>
<tr>
<td></td>
<td>ResNet</td>
<td>0.739†</td>
<td>0.733</td>
<td>0.223</td>
<td>0.133†</td>
<td>0.542</td>
<td>0.396</td>
</tr>
<tr>
<td>Pre-trained PLM</td>
<td>ProtBert</td>
<td>0.794†</td>
<td>-</td>
<td>-</td>
<td>0.634†</td>
<td>0.866</td>
<td>0.308</td>
</tr>
<tr>
<td></td>
<td>ProtBert*</td>
<td>0.209†</td>
<td>0.507</td>
<td>0.277</td>
<td>0.123†</td>
<td>0.619</td>
<td>0.164</td>
</tr>
<tr>
<td></td>
<td>ESM-1b</td>
<td>0.821†</td>
<td>-</td>
<td>-</td>
<td>0.704†</td>
<td>0.878</td>
<td>0.386</td>
</tr>
<tr>
<td></td>
<td>ESM-1b*</td>
<td>0.454†</td>
<td>0.573</td>
<td>0.241</td>
<td>0.337†</td>
<td>0.605</td>
<td>0.178</td>
</tr>
<tr>
<td></td>
<td>ESM-1v*</td>
<td>0.533</td>
<td>0.580</td>
<td>0.171</td>
<td>0.359</td>
<td>0.632</td>
<td>0.180</td>
</tr>
<tr>
<td></td>
<td>ESM-2*</td>
<td>0.475</td>
<td>0.581</td>
<td>0.190</td>
<td>0.422</td>
<td>0.632</td>
<td>0.189</td>
</tr>
<tr>
<td>GNN-based Methods</td>
<td>GCN</td>
<td>0.824</td>
<td>0.730</td>
<td>0.361</td>
<td>0.745</td>
<td>0.865</td>
<td>0.466</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>0.821</td>
<td>0.741</td>
<td>0.369</td>
<td>0.757</td>
<td>0.873</td>
<td>0.508</td>
</tr>
<tr>
<td></td>
<td>Graph. Transf.</td>
<td>0.827</td>
<td>0.749</td>
<td>0.389</td>
<td>0.753</td>
<td>0.876</td>
<td>0.548</td>
</tr>
<tr>
<td></td>
<td>NodeFormer</td>
<td>0.822</td>
<td>0.744</td>
<td>0.393</td>
<td>0.755</td>
<td>0.877</td>
<td>0.543</td>
</tr>
<tr>
<td>Ours</td>
<td>EVOLMPNN</td>
<td>0.835</td>
<td>0.757</td>
<td>0.433</td>
<td>0.768</td>
<td>0.881</td>
<td>0.584</td>
</tr>
</tbody>
</table>

EvolMPNN outperforms all baselines on 8 of 9 splits. Table 2 summarises performance comparison on AAV, GB1 and Fluorescence datasets. EvolMPNN achieves new state-of-the-art performance on most splits of three datasets, with up to 6.7% improvements to baseline methods. This result vindicates the effectiveness of our proposed design to capture evolution information for homologous protein property prediction. Notably, GNN-based methods that utilise manually constructed graph structure do not enter top-2 on 9 splits and two Transformer structure models, i.e., GraphTransformer and NodeFormer, often outperform such methods. It can be understood since homology graph construction is a challenging biomedical task (Pearson, 2013), the simple $K$-NN graph construction is not an effective solution.

Large-scale PLM models are dominated by simple models. Surprisingly, we find that smaller models, such as CNN and ResNet, can outperform large ESM variants pre-trained on million- and billion-scale sequences. For instance, ESM-1v has about 650 million parameters and is pre-trained on around 138 million UniRef90 sequences (Meier et al., 2021). Yet, CNN outperforms ESM-1v on three splits of Fluorescence dataset. This indicates the necessity of designs targeting specifically the crucial homologous protein engineering task.

Our proposed extension models outperform all baselines on GB1 dataset. We performed additional experiments on GB1 datasets to investigate the performance of two extended models, i.e., EVOLGNN and EVOLFORMER and study the influence of different residue embedding initialisation methods. The results summarised in Table 3 evince that EVOLMPNN outperforms the other two variants in three splits, and all our proposed models outperform the best baseline. This result confirms the effectiveness of encoding evolution information for homologous protein property prediction. Besides, the models adopting the PLM encoding $\phi_{PLM}$ achieve better performance than.
Table 3: Results on GB1 datasets (metric: Spearman’s $\rho$) of our proposed methods, with different residue embeddings. Top-2 performances of each split marked as **bold** and *underline*.

<table>
<thead>
<tr>
<th>Model</th>
<th>Split</th>
<th>2-VS-R.</th>
<th>3-VS-R.</th>
<th>L.-VS-H.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Baseline</td>
<td></td>
<td>0.757</td>
<td>0.878</td>
<td>0.548</td>
</tr>
<tr>
<td>EVOLMPNN ($\Phi^{\text{OH}}$)</td>
<td></td>
<td>0.766</td>
<td>0.877</td>
<td>0.553</td>
</tr>
<tr>
<td>EVOLGNN ($\Phi^{\text{OH}}$)</td>
<td></td>
<td>0.764</td>
<td>0.866</td>
<td>0.536</td>
</tr>
<tr>
<td>EVOLFORMER ($\Phi^{\text{OH}}$)</td>
<td></td>
<td>0.764</td>
<td>0.868</td>
<td>0.537</td>
</tr>
<tr>
<td>EVOLMPNN ($\Phi^{\text{PLM}}$)</td>
<td></td>
<td>0.768</td>
<td>0.881</td>
<td>0.584</td>
</tr>
<tr>
<td>EVOLGNN ($\Phi^{\text{PLM}}$)</td>
<td></td>
<td>0.767</td>
<td>0.879</td>
<td>0.581</td>
</tr>
<tr>
<td>EVOLFORMER ($\Phi^{\text{PLM}}$)</td>
<td></td>
<td>0.766</td>
<td>0.879</td>
<td>0.575</td>
</tr>
</tbody>
</table>

those using the one-hot encoding $\Phi^{\text{OH}}$. From this experiment, we conclude that residue information provided by PLM helps to capture protein’s evolution information.

### 5.2 Analysis of Performance

The performance of EVOLMPNN comes from its superior predictions on high mutational mutants. For the LOW-VS-HIGH split of GB1 dataset, we group the test proteins into 4 groups depending on their number of mutations. Next, we compute three models, including EVOLMPNN, ESM-1b (fine-tuned PLM model) and CNN (best baseline), prediction performances on each protein group and present the results in Figure 3. EVOLMPNN outperforms two baselines in all 4 protein groups. Notably, by demonstrating EVOLMPNN’s clear edge in groups of no less than 3 mutations, we confirm the generalisation effectiveness from low mutational mutants to high mutational mutants. As per inference time, EVOLMPNN and CNN require similar inference time ($\approx 5$ secs), 36× faster than ESM-1b ($\approx 3$ mins).

Influence of hyper-parameter settings on EVOLMPNN. We present in Figure 4 a group of experiments to study the influence of some hyper-parameters on EVOLMPNN, including the number of attention heads, embedding dimension and the number of layers of residue encoder and evolution encoder. EVOLMPNN exhibits stable performance on different hyper-parameter settings.

### 6 Conclusion and Future Work

We propose Evolution-aware Message Passing Neural Network (EVOLMPNN), that integrates both protein sequence information and evolution information by means of residues to predict the mutational effect on homologous proteins. Empirical and theoretical studies show that EVOLMPNN and its extended variants (EVOLGNN and EVOLFORMER) achieve outstanding performance on several benchmark datasets while retaining reasonable computation complexity. In future work, we intend to incorporate 3D protein structure information towards general-purpose homologous protein models.
REFERENCES


A ALGORITHM

Algorithm 1: The framework of EVOLMPNN

Input: Protein set \( \mathcal{M} = \{P_i\}_{i=1,2,\ldots,M} \) and each protein sequence \( P_i \) contains a residue set \( \{r_j\}_{j=1,2,\ldots,N} \); Message computation function \( F_{\text{MESSAGE}} \) that outputs an \( d \) dimensional message; \( \text{COMBINE}(\cdot) \) and \( \text{CONCAT}(\cdot) \) operators.

Output: Protein embeddings \( \{Z_i\}_{i=1,2,\ldots,M} \)

```
1 \( H_i \leftarrow \text{PLMENCODER}(P_i) \)
2 \( X_i \leftarrow \phi^{\text{OH}}(\{r_j\}_{j=1,2,\ldots,N}) / \phi^{\text{PLM}}(\{r_j\}_{j=1,2,\ldots,N}) \)
3 \( X_i \leftarrow X_i \odot \phi^{\text{POS}} \)
4 \( R_i^{(0)} \leftarrow X_i \)
5 for \( \ell = 1, 2, \ldots, L_r \) do
6   for \( i = 1, 2, \ldots, N \) do
7     \( R_i^{(\ell)} \leftarrow \text{NODEFORMER}(R_i^{(\ell-1)}) \)
8 end
9 end
10 \( R_i \leftarrow R_i^{(L_r)} \)
11 \( H_i^{(0)} \leftarrow H_i \)
12 for \( \ell = 1, 2, \ldots, L_p \) do
13   \( \{S_j\}_{j=1,2,\ldots,k} \sim \mathcal{M} \)
14   for \( i = 1, 2, \ldots, M \) do
15     for \( j = 1, 2, \ldots, k \) do
16       \( d_{ij} = \text{COMBINE}(R_i - R_{S_j}) \)
17     end
18     \( H_i^{(\ell)} = \text{COMBINE}([F_{\text{MESSAGE}}(i, j, H_j^{(\ell)}, d_{ij})]_{j=1,2,\ldots,k}) \)
19     \( H_i^{(\ell+1)} = \text{CONCAT}(H_i^{(\ell)}, H_j^{(\ell)}W^{\ell}) \)
20 end
21 \( Z_i = H_i^{(L_p)} \)
22 \( Z^R = \text{MEAN POOL}(R_i) \)
23 \( Z_i = \text{CONCAT}(Z_i^P, Z_i^R) \)
```

We summarise the process of Evolution-aware Message Passing Neural Network (EVOLMPNN) in Algorithm 1. Given a protein set \( \mathcal{M} = \{P_i\}_{i=1,2,\ldots,M} \) and each protein sequence \( P_i \) contains a residue set \( \{r_j\}_{j=1,2,\ldots,N} \). For each protein \( P_i \), we first initialise protein sequence \( (H_i) \) and residue embeddings \( (X_i) \) (line 1-2). After, the residue embeddings are empowered with positional encoding \( (\Phi^{\text{POS}}) \) to get \( X_i \) (line 3). Such a design will help us to record the position of a mutation occurring in the protein sequence in the following steps. Then, we update residue embedding based on a contact map, which records the chemical reactions between residues after folding into a 3D structure (line 4-10). Furthermore, we aggregate evolution-aware embeddings by means of updated residue embeddings (line 11-17) and integrate them with protein sequence embeddings to empower them with evolution-aware semantics (line 18-21). Finally, we merge protein sequence and residue embeddings as the final protein embeddings, which contain comprehensive information and make predictions based on them (line 22-23).

B THEORETICAL ANALYSIS

Inspired by (You et al., 2019), we adopt Bourgain’s Theorem (Bourgain, 1985) to guide the random anchor number \( k \) of the evolution encoding layer, such that the resulting embeddings are guaranteed to have low distortion. Specifically, distortion measures the faithfulness of the embeddings in preserving distances (in our case, is the differences between protein sequences on a homology network) when mapping from one metric space to another metric space, which can be defined as:
Definition 1 (Distortion). Given two metric space \((M, F_{D\text{ist}})\) and \((Z, F'_{D\text{ist}})\) and a function \(f: M \to Z\), \(f\) is said to have distortion \(\alpha\), if \(\forall P_i, P_j \in M, \frac{1}{\alpha} F_{D\text{ist}}(P_i, P_j) \leq F'_{D\text{ist}}(f(P_i), f(P_j)) \leq \alpha F_{D\text{ist}}(P_i, P_j)\).

Theorem 1 (Bourgain Theorem). Given any finite metric space \((M, F_{D\text{ist}})\), with \(|M| = M\), there exists an embedding of \((M, F_{D\text{ist}})\) into \(\mathbb{R}^k\) under any \(l_p\) metric, where \(k = O(\log^2 M)\), and the distortion of the embedding is \(O(\log M)\).

Theorem 1 states the Bourgain Theorem (Bourgain 1985), which shows the existence of a low distortion embedding that maps from any metric space to the \(l_p\) metric space.

Theorem 2 (Constructive Proof of Bourgain Theorem). For metric space \((M, F_{D\text{ist}})\), given \(k = \log^2 M\) random sets \(\{S_j\}_{j=1,2,\ldots,\log^2 M} \subset M\), \(S_j\) is chosen by including each point in \(M\) independently with probability \(\frac{1}{2^j}\). An embedding method for \(P_i \in M\) is defined as:

\[
  f(P_i) = \left( \frac{F_{D\text{ist}}(P_i, S_1)}{k}, \frac{F_{D\text{ist}}(P_i, S_2)}{k}, \ldots, \frac{F_{D\text{ist}}(P_i, S_{\log^2 M})}{k} \right),
\]

Then, \(f\) is an embedding method that satisfies Theorem 1.

Anchor Selection. EVOLMPNN can be viewed as a generalisation of the embedding method of Theorem 2 (Linial et al. 1995), where \(F_{D\text{ist}}(\cdot)\) is generalised via message passing functions (Eq 3, Eq 6). Therefore, Theorem 2 offers a theoretical guide that \(O(\log^2 M)\) anchors are needed to guarantee low distortion embedding. Following this principle, EVOLMPNN choose \(k = \log^2 M\) random anchors, denoted as \(\{S_j\}_{j=1,2,\ldots,\log^2 M}\), and we sample each protein in \(M\) independently with probability \(\frac{1}{2^j}\).

C Datasets

Adeno-associated virus (AAV) capsid proteins are responsible for helping the virus carrying viral DNA into a target cell (Vandenbergh et al. 2009); there is great interest in engineering versions of these proteins for gene therapy (Bryant et al. 2021; Büning et al. 2015; Barnes et al. 2019). (Bryant et al. 2021) produces mutants on a 28 amino-acid window from position 561 to 588 of VP-1 and measures the fitness of resulting variants with between 1 and 39 mutations. We adopt three splits from the benchmark (Dallago et al. 2021), including 2-VS-REST, 7-VS-REST and LOW-VS-HIGH.

GB1 is the binding domain of protein G, an immunoglobulin binding protein found in Streptococcal bacteria (Sauer-Eriksson et al. 1995; Sjöbring et al. 1991). Wu et al. (2016) measure the fitness of generated mutations. We adopt three splits from the benchmark (Dallago et al. 2021), including 2-VS-REST, 3-VS-REST and LOW-VS-HIGH.

The green fluorescent protein is an important marker protein, enabling scientists to see the presence of the particular protein in an organic structure by its green fluorescence (Tsien 1998; Sarkisyan et al. 2016) assess the fitness of green fluorescent protein mutants. We adopt one available split, 3-VS-REST, from the benchmark (Xu et al. 2022). Besides, in order to evaluate the models’ effectiveness, we add two splits, 2-VS-REST and LOW-VS-HIGH, in this paper.

D Baselines

We present details about our baseline and proposed models in Table 4.
Table 4: Description and implementation of baseline methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Encoder</th>
<th>Pooling</th>
<th>Output layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levenshtein</td>
<td>Levenshtein distance to wild-type.</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BLOSUM62</td>
<td>BLOSUM62-score relative to wild-type.</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DDE</td>
<td>Peptide Deviation from Expected Mean</td>
<td>2-layer MLP</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Moran</td>
<td>Moran correlation</td>
<td>2-layer MLP</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSTM</td>
<td>Simple LSTM model</td>
<td>3-layer LSTM</td>
<td>Weighted sum pool</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>Transformer</td>
<td>Simple Transformer model</td>
<td>4-layer Transformer, 4 attention heads</td>
<td>-</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>CNN</td>
<td>Simple convolutional model</td>
<td>2-layer CNN</td>
<td>Max pool</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>ResNet</td>
<td>Classic framework of skip connections and residual blocks</td>
<td>8-layer ResNet</td>
<td>Attentive weighted sum</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>ProtBert</td>
<td>750M param transformer pre-trained on Uniref 50</td>
<td>30-layer BERT 16 attention heads</td>
<td>Linear pool</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>ESM-1b</td>
<td>650M param transformer pre-trained on Uniref 50</td>
<td>-</td>
<td>Mean pool</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>ESM-1v</td>
<td>650M param transformer pre-trained on Uniref 90</td>
<td>-</td>
<td>Mean pool</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>ESM-2</td>
<td>3B param transformer pre-trained on Uniref 50</td>
<td>-</td>
<td>Mean pool</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>GCN</td>
<td>Graph convolutional network</td>
<td>2/3-layer GCN encoder</td>
<td>-</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>GAT</td>
<td>Graph attention network</td>
<td>2/3-layer GAT encoder</td>
<td>-</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>GraphTransf.</td>
<td>Transformer model designed for graphs</td>
<td>2/3-layer GraphTransf. encoder</td>
<td>-</td>
<td>1-layer MLP</td>
</tr>
<tr>
<td>Nodeformer</td>
<td>Efficient Transformer variant design for graphs</td>
<td>2/3/4-layer Nodeformer encoder</td>
<td>-</td>
<td>1-layer MLP</td>
</tr>
<tr>
<td>EVOLMPNN</td>
<td>-</td>
<td>2/3-layer Residue encoder and Evolution encoder</td>
<td>-</td>
<td>1-layer MLP</td>
</tr>
<tr>
<td>EVOLGNN</td>
<td>-</td>
<td>2/1-layer Residue encoder and Evolution encoder</td>
<td>-</td>
<td>1-layer MLP</td>
</tr>
<tr>
<td>EVOLFORMER</td>
<td>-</td>
<td>2/3-layer Residue encoder and Evolution encoder</td>
<td>-</td>
<td>1-layer MLP</td>
</tr>
</tbody>
</table>