Supplementary Material: ADOPT: intrinsic protein disorder prediction through deep bidirectional transformers

Istvan Redl, Carlo Fisicaro, Oliver Dutton, Falk Hoffmann, Louie Henderson, Benjamin M.J. Owens, Matthew Heberling, Emanuele Paci, Kamil Tamiola *

1 APPENDIX A

1.1 Transformer

The Transformer encoder model of ESM is a multi-layer bidirectional Transformer encoder architecture derived from the original implementation[1]. The ESM uses Bidirectional Encoder Representations from BERT-like architecture [2, 3], which alleviates the undirectionality constraint related to a left-to-right architecture where every token can only attend to previous tokens in the self-attention layers of the Transformer. The ESM utilizes a masked language model [4] in which some of the tokens from the input are randomly masked with the objective of predicting the original vocabulary id of the masked residue based only on its context.

1.1.1 Positional encoding

Here $\mathbf{x}$ is mapped into input the embedding matrix $E \in \mathbb{R}^{n \times d_{mod}}$, where $E_{i,*}$ is the embedding vector $\mathbf{e}_i \in \mathbb{R}^{d_{mod}}$ of x_i for $i = 1, 2, \dots, n$.

The vector $\mathbf{e}_i$ is defined by the affine transformation, also called linear projection, $\mathbf{e}_i = \mathbf{o}_i W^T + \mathbf{b}$ [1], where W^T indicates the transpose of W, $\mathbf{o}_i \in \{0,1\}^v$ is the *one-hot encoded* representation of x_i so that $v = |V|$ is the cardinality of the set V while $W \in \mathbb{R}^{d_{mod} \times v}$ and $\mathbf{b} \in \mathbb{R}^{d_{mod}}$ are respectively the weight matrix and the bias computed during the training procedure. The embedding layer is defined as $E = \mathit{embedding}(\mathbf{x})$.

Since the Transformer contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, one must inject some information about the relative or absolute position of the amino acid residues in the protein sequence. To this end, a "positional encodings" $\mathbf{c}_i$ is added to the input embedding $\mathbf{e}_i$. The ESM makes use of learned positional embeddings [3] i.e an embedding layer fed with the position of each residue in place of the residue itself.

Finally, stacking $\mathbf{d}_i = \mathbf{e}_i + \mathbf{c}_i$ for $i = 1, 2, \dots, n$ one gets the matrix $D \in \mathbb{R}^{n \times d_{mod}}$.

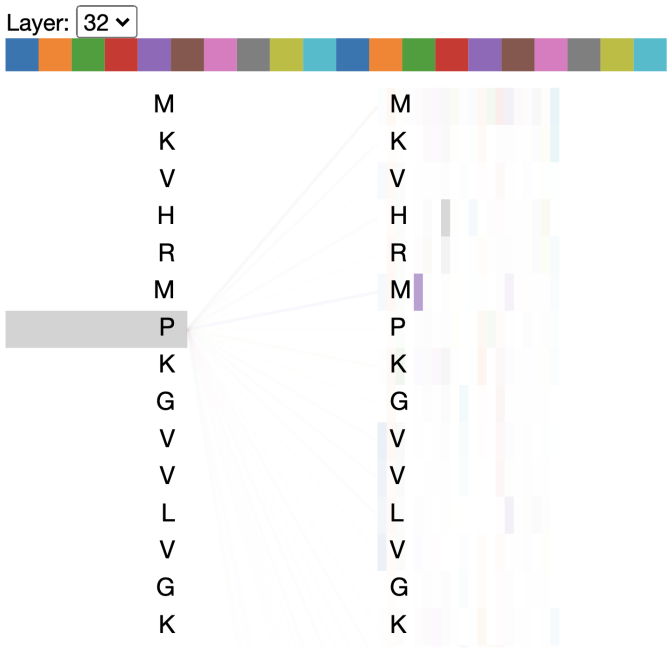
1.1.2 Attention mechanism

After applying a layer normalization [5], once gets $N_{\mathrm{D}} = \mathit{layerNorm}(D)$, where $\mathit{layerNorm}(M)$ stands for the *layerNorm function* applied on each column of the matrix M and $N_{\mathrm{D}} \in \mathbb{R}^{n \times d_{mod}}$.

Three different linear projections are then applied to N_{D} getting respectively, except for the bias term, the *query* matrix $N_{\mathrm{D}} W^Q = Q \in \mathbb{R}^{n \times d_{mod}}$, the *key* matrix $N_{\mathrm{D}} W^K = K \in \mathbb{R}^{n \times d_{mod}}$ and the *value* matrix $N_{\mathrm{D}} W^V = V \in \mathbb{R}^{n \times d_{mod}}$ with $W^Q, W^K$and $W^V \in \mathbb{R}^{d_{mod} \times d_{mod}}$ as weight matrices, respectively, computed at training time.

Each of the matrices $Q, K$and V is then reshaped into h different matrices i.e. $Q_j \in \mathbb{R}^{n \times d_k}$, $K_j \in \mathbb{R}^{n \times d_k}$ and $V_j \in \mathbb{R}^{n \times d_v}$ where $j = 1, 2, \dots, h \in \mathbb{N}^+$ and $d_k = d_v = \frac{d_{mod}}{h}$.

*To whom correspondence should be addressed. Tel: +39 338 592 0947; Email: kamil@peptone.io



Supplementary Figure S1: Last multi-attention layer in the **ESM-1b** Transformer model related to a singular protein entry in the CheZod “1325”, identified with the 25096 index. Lines depict the attention from each token (left) to every other token (right). Darker lines indicate higher attention weights, whereas the colours denote different attention heads. The “Layer” drop-down indicates the model layer (zero-indexed). We cut the sequence for visualisation constraints.

The j-th matrices $Q_j, K_j$and V_j are then fed into the *scaled dot-product attention* layer

$$A_j = \sigma\left(\frac{Q_j K_j^T}{\sqrt{d_k}}\right) V_j$$

where $\sigma(M)$ stands for the *softmax function* [6] applied on each column of the matrix M. The matrix A_j is called *attention head* j with $A_j \in \mathbb{R}^{n \times d_v}$ $\forall j \in \{1, 2, \ldots, h\}$ and h represents the number of attention heads employed.

The attention heads are then concatenated into $A = (A_1, A_2, \ldots, A_h)$ with $A \in \mathbb{R}^{n \times h d_v} = \mathbb{R}^{n \times d_{mod}}$ and a linear projection $A_{\mathrm{MH}} = AW^O$ is applied where, $W^O \in \mathbb{R}^{d_{mod} \times d_{mod}}$ and $A_{\mathrm{MH}} \equiv A_{\mathrm{MH}}(Q, K, V) \in \mathbb{R}^{n \times d_{mod}}$. The set of operations which return A_{MH} with a matrix M as input, is called $multiHeadAttention(M)$.

A *residual connection* [7] is then employed, getting $R = identity(N_{\mathrm{D}} + A_{\mathrm{MH}})$ where $identity(M)$ stands for the *identity operator* [2] applied on the matrix M and $R \in \mathbb{R}^{n \times d_{mod}}$. See Supplementary Figure S1 for a multi-head attention output visualisation, related to the CheZod dataset.

1.1.3 Feed-forward network

A *layer normalization* is then applied to the multi-head attention output, so that one gets $N = layerNorm(R)$ with $N \in \mathbb{R}^{n \times d_{mod}}$.

The matrix N is then fed into the *position independent feed-forward network*

$$\mathbf{f}_i = g(\mathbf{n}_i W_1^i + \mathbf{b}_1^i)W_2^i + \mathbf{b}_2^i$$

where $\mathbf{n}_i \equiv N_{i,*}$ and $g(\mathbf{m})$ is the *Gaussian error linear unit* activation function [8] applied to a vector $\mathbf{m}$; $W_1^i \in \mathbb{R}^{d_{mod} \times d_f}$ and $W_2^i \in \mathbb{R}^{d_f \times d_{mod}}$ are the weight matrices whereas $\mathbf{b}_1^i \in \mathbb{R}^{d_f}$ and $\mathbf{b}_2^i \in \mathbb{R}^{d_{mod}}$ are the bias terms $\forall i \in \{1, 2, \ldots, n\}$. Here $d_f = 4d_{mod}$ for convenience while stacking the position independent feed-forward networks $\mathbf{f}_i$ for $i = 1, 2, \ldots, n$ one gets $F \equiv F(N) \in \mathbb{R}^{n \times d_{mod}}$. The set of operations which return F with a matrix M as input is called, $feedForwardNetwork(M)$.

Finally, *mutatis mutandis*, a *residual connection* is employed, getting $R_\text{F} = identity(N + F)$ with $R_\text{F} \in \mathbb{R}^{n \times d_{mod}}$. Note that the ESM makes use of *pre-activation blocks*, where the layer normalization is applied prior to the activation and no *dropout* [9] is used.

1.1.4 Encoder block

Once the *positional encoding* layer is applied you get the residue level representation D which is then fed into the a *layer normalization* followed by the *multi-head attention* to which a *residual connection* is applied, yielding the residual matrix R. The matrix R is then fed into another *layer normalization* followed by the *position independent feed-forward network* to which another *residual connection* is applied, yielding R_F.

Therefore the *encoder block* layer can be defined as

$$\begin{aligned}
&encoder(D): \\
&\qquad N_\text{D} \leftarrow layerNorm(D) \\
&\qquad A_\text{MH} \leftarrow multiHeadAttention(N_\text{D}) \\
&\qquad R \leftarrow identity(D + A_\text{MH}) \\
&\qquad N \leftarrow layerNorm(R) \\
&\qquad F \leftarrow feedForwardNetwork(N) \\
&\qquad R_\text{F} \leftarrow identity(N + F)
\end{aligned}$$

Note that the residue level representation, input matrix D of the *encoder block* has the same dimension of the output matrix R_F of the same block.

1.1.5 Loss

The Transformer has been trained using the *masked language modeling* objective [3] where the input $\mathbf{x}$ is corrupted by replacing a fraction of the residues with a special mask token "`<mask>`" and the expectation $\mathbb{E}$ is computed on the set of masked indices at first, and then on the set of protein sequences. The network has been trained to predict the missing tokens from the corrupted sequence, which results in the *minimisation* of:

$$\mathcal{L}_\text{MLM} = \mathbb{E}_{\mathbf{x} \sim \mathbf{X}} \mathbb{E}_T \sum_{t \in T} -log \left[p \left(x_t | \mathbf{x}_{/T} \right) \right],$$

where, for each sequence $\mathbf{x}$ extracted from the vector of *random variables* $\mathbf{X}$, a set of indices T is sampled to mask, replacing the true residue x_t with the mask token whereas $\mathbf{x}_{/T}$ represents the masked sequence i.e the context of x_t.

The method implemented in [2] has a *token dropout* scheme which replaces the mask token embedding with a fixed vector of zeros so that $\mathbf{e}_t = \mathbf{0}\ \forall t \in T$ with $\mathbf{0} = \underbrace{(0, 0, \ldots, 0)}_{d_{mod} \text{ times}}$.

1.1.6 Masking

The masking strategy in [2] has been adopted from BERT [3], where 15% of the input tokens were selected and predicted through the minimisation of $\mathcal{L}_{\text{MLM}}$. Of these 80% were replaced with mask token and 10% with a random residue extracted from a uniform distribution; 10% not changed.

1.1.7 Architecture

The Transformer is composed of $l \in \mathbb{N}^+$ stacked *encoder blocks*, each fed with the output of the previous one and a final *layer normalization* applied to the output of the last layer.

The output of the last block is denoted as $Z \in \mathbb{R}^{n \times d_{mod}}$ with $\mathbf{z}_i = Z_{i,*}$ and,

$$Z = layerNorm(encoder^{[l]}(D))$$

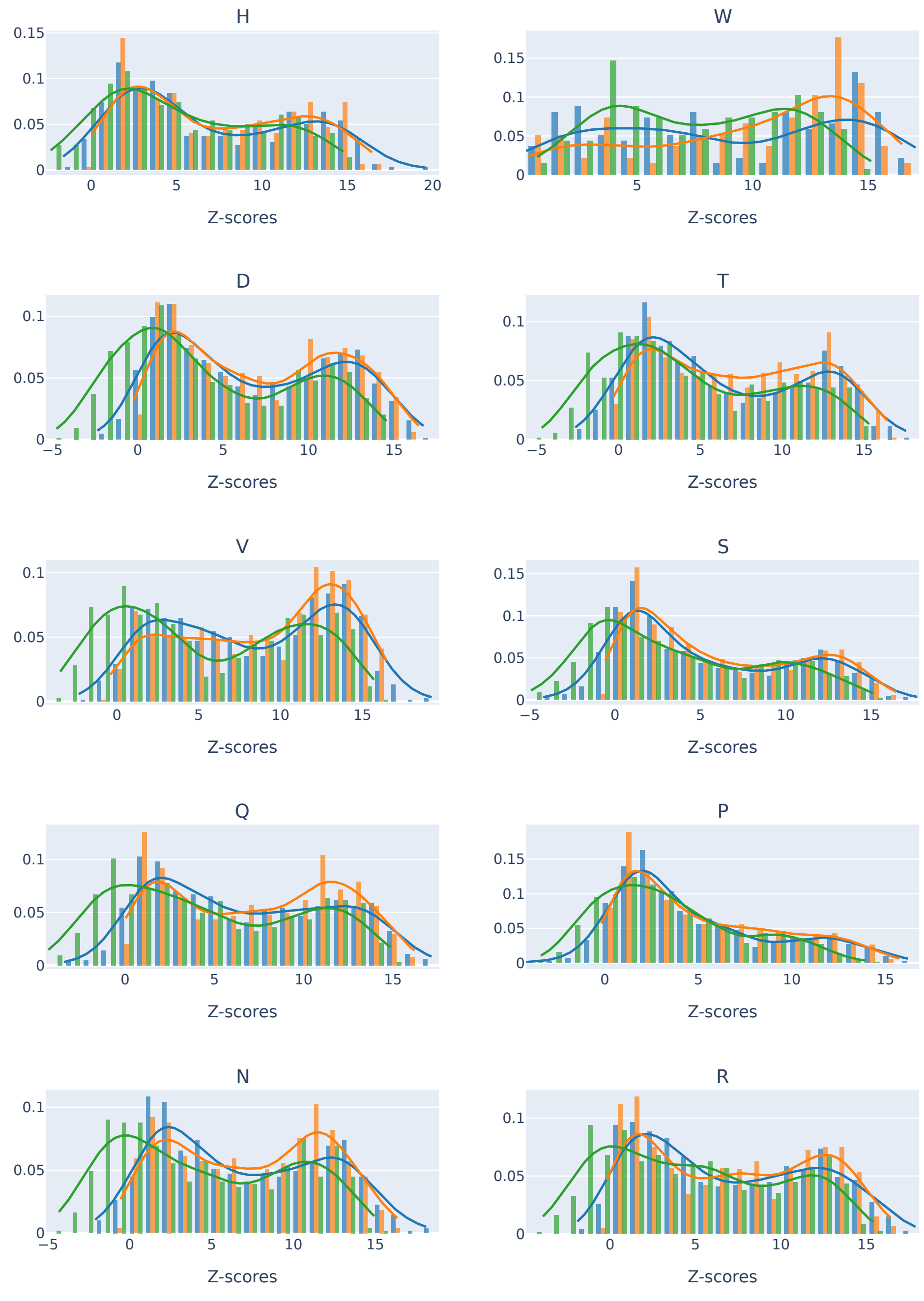
where, $f^{[l]}(M) = \underbrace{f(f \cdots f(M))}_{l \text{ times}}$.

Please, refer to [2] for additional details.

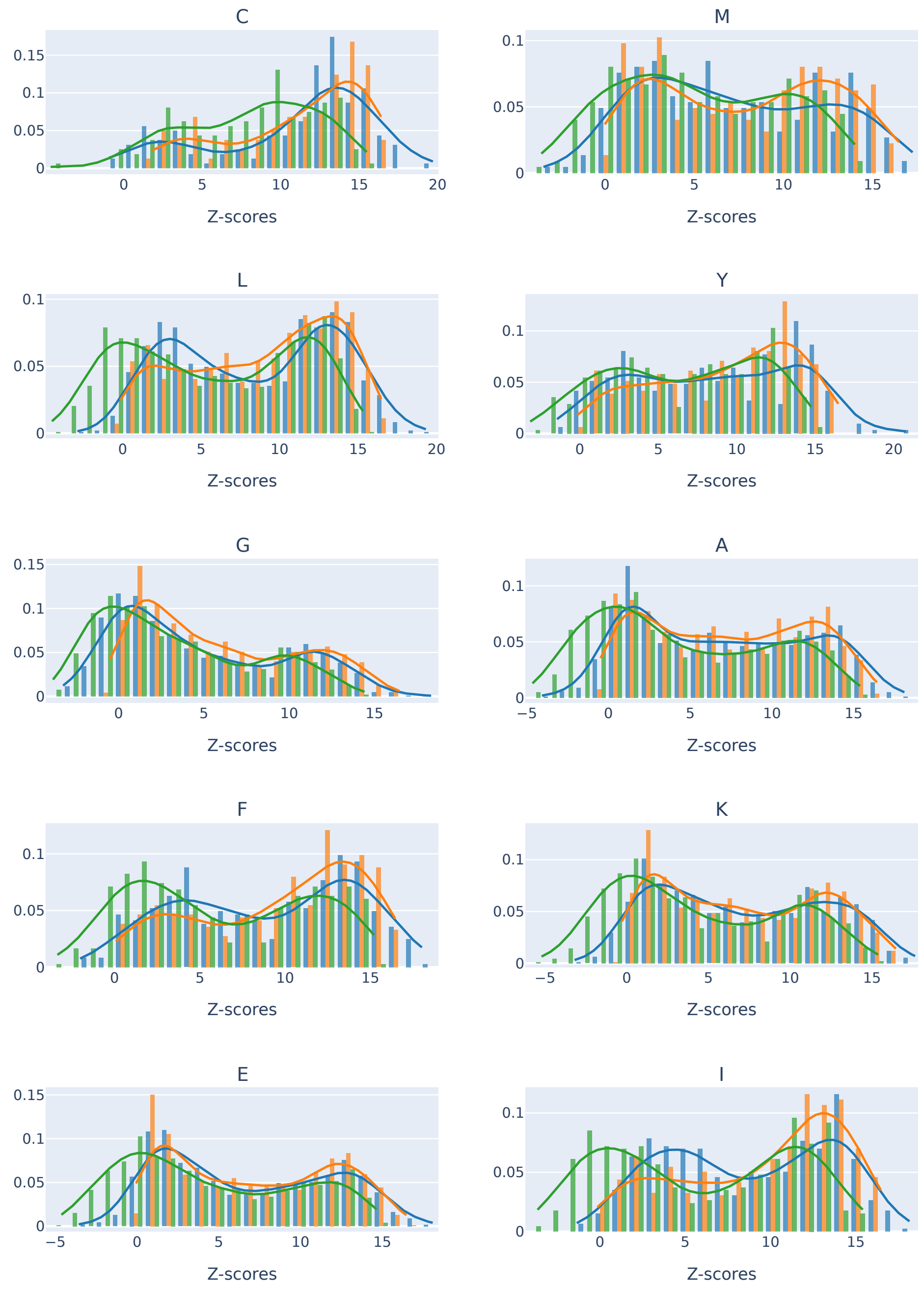
It is also noteworthy that a projection[3] back to the size of the vocabulary v has been applied to Z in order to get the log probabilities one needs to compute $\mathcal{L}_{\text{MLM}}$.

Finally, the Transformer was trained in batches of $b \in \mathbb{N}^+$ proteins, each. Therefore the input was a 2nd rank tensor $X^{\alpha,\mu}$ and each layer was applied on the batch whereas the output was a 3rd rank tensor $Z^{\alpha,\mu,\nu}$ where $\alpha \in \{1, 2, \ldots, b\}, \mu \in \{1, 2, \ldots, n\}$ and $\nu \in \{1, 2 \ldots, d_{mod}\}$.

2 APPENDIX B



Supplementary Figure S2: Residue level histograms/density plots of *actual* (**green**), *ESM-1b* (**blue**) and *ODiNPred* (**orange**) predicted Z-scores for residues (from top left to bottom right) ['H', 'W', 'D', 'T', 'V', 'S', 'Q', 'P', 'N', 'R']. While in general the ESM transformer based predictor is similar to ODiNPred, in some instances (e.g. residues 'T' and 'R') it puts more mass closer to the actual density, than ODiNPred.



Supplementary Figure S3: Residue level histograms/density plots of *actual* (**green**), *ESM-1b* (**blue**) and *ODiNPred* (**orange**) predicted Z-scores for residues (from top left to bottom right) ['C', 'M', 'L', 'Y', 'G', 'A', 'F', 'K', 'E', 'I']. While in general the ESM transformer based predictor is similar to ODiNPred, in some instances (e.g. residues 'G' and 'Y') it puts more mass closer to the actual density, than ODiNPred.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[2] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *Proceedings of the National Academy of Sciences*, vol. 118, no. 15, 2021.

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

[4] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[5] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

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

[8] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

Notes

1. The linear projection applied of the protein sequence p can be represented, except for the bias term, as $D = OW^T$ where $O \in \mathbb{R}^{n \times v}$ is the one hot encoding matrix
2. The identity operator $\hat{\mathbb{I}}$ is defined as $\hat{\mathbb{I}}M \equiv M$
3. A Gaussian error linear unit function and a layer normalisation are applied before the output