TRANSFORMERS TRAINED ON PROTEINS CAN LEARN TO ATTEND TO EUCLIDEAN DISTANCE

Isaac Ellmen[1], **Constantin Schneider**[2*], **Matthew I.J. Raybould**[1], **Charlotte M. Deane**[1†]
[1]Department of Statistics, University of Oxford
[2]Exscientia

ABSTRACT

While conventional Transformers generally operate on sequence data, they can be used in conjunction with structure models, typically SE(3)-invariant or equivariant graph neural networks (GNNs), for 3D applications such as protein structure modelling. These hybrids typically involve either (1) preprocessing/tokenizing structural features as input for Transformers or (2) taking Transformer embeddings and processing them within a structural representation. However, there is evidence that Transformers can learn to process structural information on their own, such as the AlphaFold3 structural diffusion model. In this work we show that Transformers can function independently as structure models when passed linear embeddings of coordinates. We first provide a theoretical explanation for how Transformers can learn to filter attention as a 3D Gaussian with learned variance. We then validate this theory using both simulated 3D points and in the context of masked token prediction for proteins. Finally, we show that pre-training protein Transformer encoders with structure improves performance on a downstream task, yielding better performance than custom structural models. Together, this work provides a basis for using standard Transformers as hybrid structure-language models. The code is available at: `https://github.com/Ellmen/attending-to-distance`.

1 INTRODUCTION

Background. Transformers typically operate on sequential data, however many applications of Transformers benefit from an ability to learn geometric reasoning. For instance, ESM-2 (Lin et al., 2023) demonstrates that in order to effectively predict masked tokens in protein *sequences*, the model has learned some ability to predict protein *structures*. Other tasks such as image processing or even natural language processing may benefit from an internal representation of objects in 3D space. However, it is unclear how Transformers can learn to use 3D representations to perform spatial reasoning. To this end, custom structural Transformers have been created which model data as graphs and represent distance between nodes as edge features in order to perform SE(3)-invariant attention (Ingraham et al., 2019; Fuchs et al., 2020; Liao & Smidt, 2023; Liao et al., 2023).

SE(3) invariance means that all functions of coordinates reduce to functions of relative distance. That is, for every function f of two coordinates x_1 and x_2, there exists an equivalent function g which depends only on their relative distance: $f(\vec{x_1}, \vec{x_2}) = g(|\vec{x_1} - \vec{x_2}|)$. Since relative distance in Euclidean space is defined as $|\vec{x}| = \sqrt{x^2 + y^2 + z^2}$, it may be easier to learn functions of the square of the relative distance — a linear combination of functions of the individual coordinates.

In this manuscript, we investigate how conventional Transformers can learn to approximate functions of the squared distance between points, thereby learning an approximately SE(3)-invariant measure of distance. In short, we show that "out of the box" Transformers can act as 3D structural models. Our main contributions are (1) to provide a theoretical explanation for how standard Transformers can learn to measure distance and perform structural reasoning, (2) to show that Transformers indeed learn Gaussian functions of distance and investigate efficient data augmentation methods

*Now at Xyme
†Corresponding author: `deane@stats.ox.ac.uk`

which can be used to learn SE(3), and (3) to train a protein masked token prediction model with coordinates and show that finetuning it for function prediction yields a model which outperforms structural GNNs.

2 THEORY

Here we introduce our theory for how Transformers can learn to measure Euclidean distance. Throughout this section, we assume that all coordinates are small. This can be achieved by scaling the inputs and outputs manually or via learned weights in the linear input/output maps. For more detail, see Appendix A.2.

2.1 GAUSSIAN SPATIAL ATTENTION

Consider a pre-norm Transformer (Xiong et al., 2020) operating on a sequence of embedded positions, $E(\vec{x_i})$. For simplicity, assume that the key and query embeddings for all heads are trivial ($Q = K = I_d$). Such mappings can easily be learned as long as the head dimension is at least d. Then, in the first layer of the Transformer, the attention matrix for all heads will be:

$$A_{i,j} = SM(\frac{LN(E(\vec{x_i}))\cdot LN(E(\vec{x_j}))}{\sqrt{d}}) \tag{1}$$

Where SM denotes the softmax function and LN denotes LayerNorm (Ba et al., 2016).

Our objective is to determine an embedding, E, such that the attention $A_{i,j}$ is a monotone decreasing function of the Euclidean distance between $\vec{x_i}$ and $\vec{x_j}$. In particular, if we choose E such that, for some $a, b \in \mathbb{R}$:

$$LN(E(\vec{x_i})) \cdot LN(E(\vec{x_j})) \approx -a|\vec{x_i} - \vec{x_j}|^2 + b \tag{2}$$

Then,

$$\begin{aligned} A_{i,j} &= SM(\frac{LN(E(\vec{x_i}))\cdot LN(E(\vec{x_j}))}{\sqrt{d}}) \\ &\approx SM(-\frac{a}{\sqrt{d}}|\vec{x_i} - \vec{x_j}|^2 + \frac{b}{\sqrt{d}}) \\ &= SM(-\frac{a}{\sqrt{d}}|\vec{x_i} - \vec{x_j}|^2) \end{aligned} \tag{3}$$

In particular, the unnormalized attention paid to x_j by x_i is:

$$A_{nonorm_{i,j}} \approx e^{-\frac{a}{\sqrt{d}}|\vec{x_i} - \vec{x_j}|^2} \tag{4}$$

Which is a Gaussian of the relative distance between the two points, i.e., Transformers can learn to approximate a 3D Gaussian to gate attention values and selectively attend to points nearby in 3D space. By learning individual LayerNorm gains or Q/K mappings, each head can tune the variance of this Gaussian filter, which allows each head to determine the appropriate spatial resolution for that type of information.

Below, we provide embeddings which satisfy Equation 2 and so lead to Gaussian attention filters.

2.2 SPATIAL POSITIONAL EMBEDDINGS

For simplicity, we consider the case of 1 spatial dimension and provide 4-dimensional embeddings, which satisfy Equation 2. The 3 (or n) dimensional case is similar. Consider the embeddings:

$$\begin{aligned} E_{trig}(x) &= (\sin(x), -\sin(x), \cos(x), -\cos(x)) \\ E_{lin}(x) &= (x, -x, 1, -1) \\ E_{quad}(x) &= (x, -x, 1 - \frac{x^2}{2}, \frac{x^2}{2} - 1) \end{aligned} \tag{5}$$

Here, E_{trig} is similar to the standard sinusoidal positional encoding for linear sequences (Vaswani et al., 2017). Then, E_{lin} and E_{quad} can be thought of as the first and second order approximations of E_{trig}, respectively. It can be shown (see Appendix A.3) that for all three embeddings:

$$LN(E(x_i)) \cdot LN(E(x_j)) \approx -2|x_i - x_j|^2 + 4 \tag{6}$$

Subject to $|x_i - x_j|$ being small for E_{trig} and subject to $|x_i|$ and $|x_j|$ being small for E_{lin} and E_{quad}. This surprising result for E_{lin} stems from the non-linearity of LayerNorm causing the constant terms to be locally quadratic. For more details, see Appendix A.3.2. An important consequence of this result is that simple linear embeddings of positions can still lead to an approximately 3D Gaussian filter of relative distance for attention.

Additionally, the approximation E_{quad} is better than that of E_{lin}, however the requirement to encode x^2 explicitly makes it harder for individual attention heads to rescale the positions appropriately, since heads have to learn the linear and quadratic scaling terms separately. However, in A.3 we show that ReGLU and SwiGLU activation functions (Shazeer, 2020) are capable of learning exactly quadratic functions of their input, which may allow some modern Transformers to learn positional embeddings of the form E_{quad} after the appropriate rescaling. This may be a useful benefit of GLU activation functions for models such as AlphaFold3 (Abramson et al., 2024).

3 Experiments

3.1 Simulated points

To test our theory of how Transformers learn to measure distance, we designed a Transformer encoder which is truncated such that the output is the unnormalized attention matrix for a single head. A diagram of this model is shown in Figure 1. We computed the loss as the l_1 difference between the output matrix and the matrix $A_{i,j} = e^{(\frac{-(x_i - x_j)^2}{200^2})}$. This corresponds to the prenormalized softmax of the negative square of the relative distance between points, as predicted by our theory. The data consisted of 10,000 "structures", each with five 3-dimensional points with coordinates randomly selected between 0 and 200. Unless otherwise indicated, the Transformer encoder for all experiments has three layers (is truncated at the third layer), an embedding dimension of 256, a feedforward dimension of 1,024, 8 heads, pre-normalization and ReLU activation. The models were trained with a batch size of 16 using the Adam optimizer with a peak learning rate of 4×10^{-4} which is reached after 4,000 warmup steps, and then is quadratically decayed.

As in AlphaFold3 (Abramson et al., 2024), we transform the input structures before they are passed through the model. Whenever a structure is loaded, its points are recentred, randomly rotated, and rescaled by a factor of $\frac{1}{16}$. This has two benefits. First, recentering and rescaling the points ensures that all coordinates stay relatively small, even before the embedding layer has learned an appropriate mapping. Second, recentering and randomly rotating gives the model resilience to translations and rotations which encourages it to learn a distance measure which is truly SE(3)-invariant.

Transformers can attend to $|\vec{x_i} - \vec{x_j}|^2$. In Section 2, we show that Transformers are theoretically capable of learning Gaussian functions of distance. We experimented with learning $e^{-|\vec{x_i} - \vec{x_j}|^p}$ for different powers p, ranging from 0.5 to 4, in increments of 0.5. Figure 2a shows the relationship between p and the validation loss. As expected, Transformers can learn to reproduce the $e^{-|\vec{x_i} - \vec{x_j}|^2}$ attention matrix most accurately which shows that a Gaussian is the most natural way for Transformers to learn to filter attention spatially.

Transformers need $n + 2$ embedding dimensions to learn distance in $\mathbb{R}^n$. Next, we explored how large a spatial embedding must be to learn a good approximation of distance for $\mathbb{R}^n$. For the

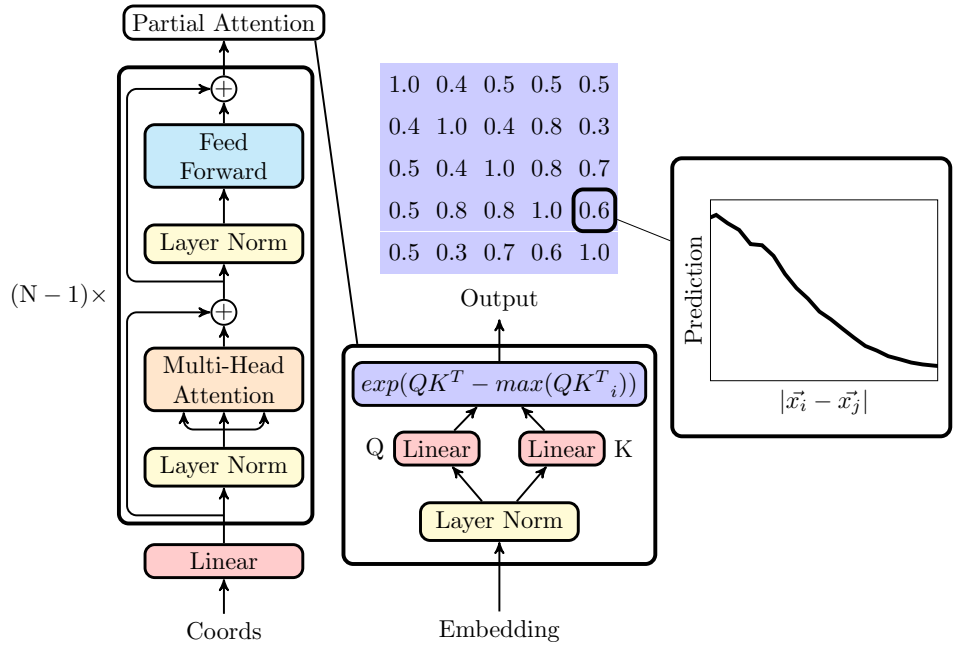
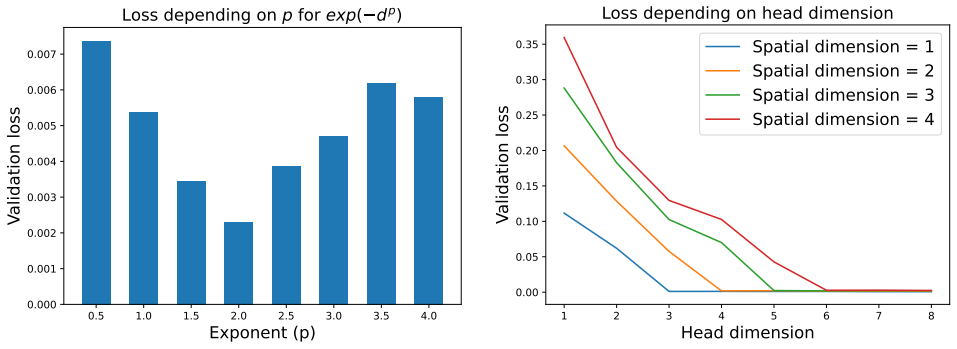


Figure 1: Overview of the simulated experiment model. Coordinates are passed through a Transformer encoder which is truncated such that the output is the unnormalized attention for a single head. Loss is computed as the difference between the attention for each pair and a Gaussian of the relative distance between those points.



(a) Loss vs exponent

(b) Loss vs head dimension

Figure 2: (a) Validation loss as a function of the exponent p. The loss is lowest for $p = 2$ which corresponds to learning a Gaussian function of distance. (b) Validation loss as a function of head dimension for different spatial dimensions. Models need a head dimension of $n + 2$ to accurately measure distance in $\mathbb{R}^n$.

case of 1 spatial dimension, we provide a theoretical 4-dimensional embedding. We trained models to predict Euclidean distance in $\mathbb{R}^n$ for $n \in 1, 2, 3, 4$ using head dimensions from 1 to 8. Figure 2b shows the relationship between validation loss and head dimension. We found that the model only needs $n + 2$ head dimensions to learn a good approximation of distance for all spatial dimensions. This means that to learn a good approximation of distance in $\mathbb{R}^3$, a model must reserve at least 5 embedding dimensions for each head. This is a surprisingly compact requirement which should be easily accommodated even in small Transformers.

3.2 PROTEINS

Proteins are a natural fit for structural Transformers because they are composed of linear sequences embedded in 3D space. As such, their properties depend on both sequential and structural features. We considered two tasks in protein modelling: predicting masked tokens as a pretraining objective and predicting protein function conditioned on embeddings generated by pretrained models. For all protein experiments we used the GO PDB dataset from DeepFRI (Gligorijević et al., 2021) which comprises ∼36K protein chains.

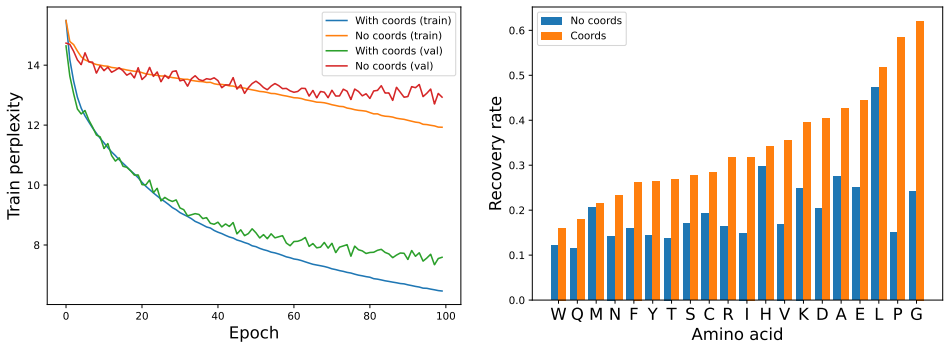
Pretraining a structural protein language model. To test a Transformer's ability to learn useful structural patterns in proteins, we trained an ESM/BERT-style (Rives et al., 2021; Devlin et al., 2019) model to complete masked token prediction. We trained two models: one with coordinates ('coords model') and the other without ('non-coords model'). The version with coordinates added a linear embedding of the coordinates to the token embedding in the same way as the simulated experiments. Both models were very similar to the smallest publicly released ESM1 model, consisting of a 6-layer Transformer encoder with a hidden dimension of 768, 12 attention heads, a feedforward dimension of 2048, and GeLU activation (Hendrycks & Gimpel, 2023). As in ESM, we omitted dropout. To prevent the model from focusing too much on linear positional information, we used Sinusoidal Positional Encodings rather than Rotary Positional Encodings (Su et al., 2022). As is common in masked token prediction, we masked 15% of tokens. Of the masked tokens, 80% were replaced with a [MASK] token, 10% were replaced with a random amino acid, and 10% were left unchanged. We clustered the data by 50% sequence identity using MMSeqs2 (Steinegger & Söding, 2017) and randomly held out 1% of the clusters to use as a validation set. We trained each model for 100 epochs with a fixed batch size of 24, resulting in approximately 150K updates. We used the Adam optimizer with 4,000 warmup steps to a peak learning rate of 2.3×10^{-4}, followed by inverse square decay. Each time a structure was loaded, its coordinates were recentred, randomly rotated, and rescaled.

As shown in Figure 3a, adding coordinates substantially improved the model, leading to a final training perplexity of 6.5 with coordinates vs 11.9 without. The final training loss for the version without coordinates (after 100 epochs) was surpassed by the version with coordinates after 8 epochs. Additionally, the final validation loss was surpassed after only 4 epochs which may indicate that the structural features learned early in training are more robust to dissimilarity in sequence space.

We also investigated the difference in sequence recovery rates between the two models. The total sequence recovery rate was ∼23% for the non-coords model compared to ∼38% for the coords model. Figure 3b shows a breakdown of the recovery rates per amino acid type. The recovery rate for the coords model was greater than or equal to that of the non-coords model for all amino acid types. The difference was particularly stark for glycine and proline, which may be related to their distinct backbone conformational preferences (Ho & Brasseur, 2005; Beck et al., 2008).

Pretrained models learn to measure distance. In the model trained in the previous paragraph, there are three inputs to the model for each token: amino acid type, sequential position, and 3D coordinates. To test if the pretrained model with coordinates was learning to measure distance as predicted, we plotted the average attention paid by each pair of tokens across all heads in a layer as a function of distance. We also plotted the average attention paid to each token as a function of relative sequence distance. To isolate the effect of each feature, we fixed all amino acids to alanine. We also fixed the linear sequence index to a constant value for all tokens while measuring 3D dependence and fixed the 3D position to $(0, 0, 0)$ while measuring linear dependence. For the distance measurements, we rounded each pairwise distance to the nearest Angstrom and computed the average for each distance value.

Figure 4 shows the plots for 3D and linear positional dependence for all layers for both models as well as the amplitude and standard deviation of the fit Gaussian for each layer. As expected, the



(a) Pretraining loss

(b) Sequence recovery breakdown

Figure 3: (a) Masked token prediction loss per epoch. Adding coordinates to the model hugely improves training and validation accuracy, as the model quickly learns to use coordinates to measure distance and compute structural features. (b) Sequence recovery rate for each amino acid type. The height of each bar represents the recovery rate for the coords model and the orange portion is the recovery rate for the non-coords model (the coords model had at least as high a recovery rate as the non-coords model for all amino acid types). The recovery rates for glycine and proline were substantially higher in the coords model, which may correspond to identification of beta turns.

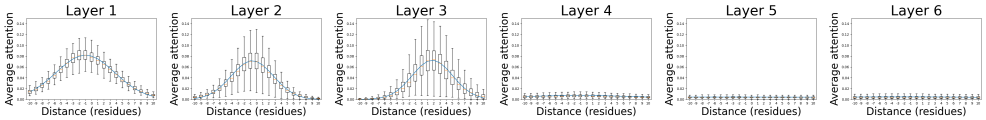
model without coordinates shows no correlation with Euclidean distance. Conversely, the model with coordinates shows a strong dependence in the early layers which is well-approximated by a Gaussian. Both models progressively widen their field of view as information passes through the layers. This may correspond to a system of reasoning where the model collects information about the local environment before processing this information alongside global patterns. We provide a similar plot without isolating each of the factors in Appendix A.5.

Protein function prediction. Finally, we tested whether the pretrained protein model embeddings could improve accuracy on a downstream task. We trained models to predict protein molecular function Gene Ontology labels (Ashburner et al., 2000). Protein function prediction has been studied extensively, and has been shown to benefit from both language and structural features (Gligorijević et al., 2021). We trained models to predict protein function based on the mean token embedding output by the pretrained models, using the same data splits as DeepFRI (Gligorijević et al., 2021). We compare our results to the DeepFRI and DeepCNN versions which were trained on PDB sequences. DeepFRI provided a useful comparison since it achieved state of the art performance on protein function prediction before models were trained using ESM embeddings. Thus we could pretrain our model on the same data as DeepFRI and evaluate the specific contributions of model architectures. DeepFRI is an ensemble of Graph Convolutional Networks (Kipf & Welling, 2022) with different propagation rules, as in Dehmamy et al. (2019). The graph structure is defined by connecting the k-nearest neighbours for each amino acid in the protein structure. DeepFRI uses a pretrained LSTM model (Graves, 2014) to generate language model embeddings for node features. DeepCNN is a Convolutional Neural Network (LeCun et al.) meant to replicate DeepGO (Kulmanov et al., 2018) but retrained on the same sequences as DeepFRI (and our model).

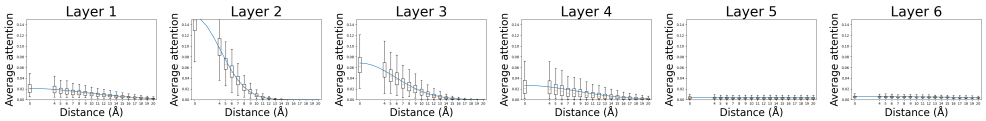
We train two models based on the pretrained models from the previous section. Our first model is based on the simple MLP model from Kulmanov et al. (2024), which is a simple 2 layer MLP block of 1,024 dimensions with a residual connection. The input to the model is a learned linear embedding of the mean token embedding output by the pretrained models.

Our second model is a finetuned version of the pretrained masked token prediction models. As in BERT (Devlin et al., 2019) the output is a learned linear projection of the final start token embedding. Each model (coords/no coords) is finetuned for 20 epochs with a constant learning rate of 3×10^{-5}.

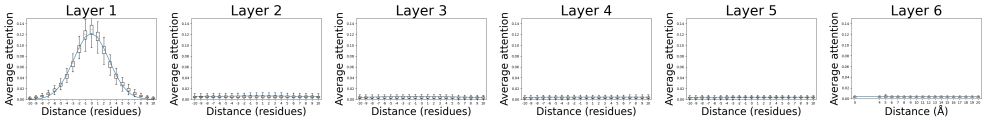
The results are shown in Table 1. Additional results for predicting biological process and cellular component are provided in Appendix A.6. Our sequence-only MLP model compared competitively with DeepCNN and our sequence-structure MLP model compared competitively with DeepFRI. The



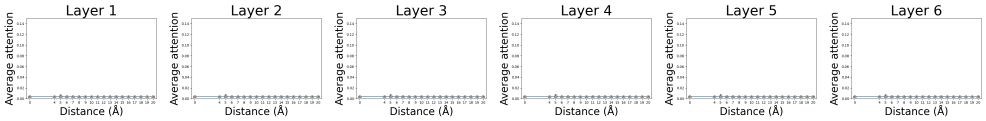
(a) Linear position, with coordinates



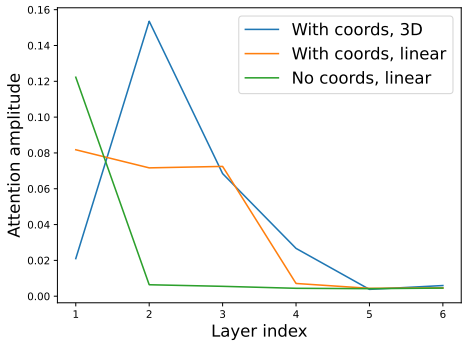
(b) 3D position, with coordinates



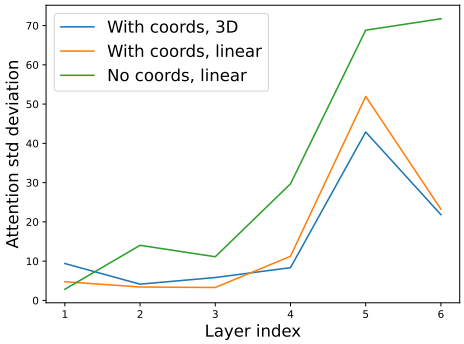
(c) Linear position, no coordinates



(d) 3D position, no coordinates



(e) Gaussian amplitudes



(f) Gaussian standard deviations

Figure 4: Average attention paid per layer to linear and 3D positional information (a-d). Fit Gaussian functions are shown in blue for each plot. The model which was trained with coordinates learns to filter heavily by linear and 3D positional information whereas the model trained without coordinates only filters by linear information in early layers. Amplitudes (e) and standard deviations (f) of the fit Gaussians show that both models pay attention to local features early and then gradually widen their fields of view.

MLP models are very simple, taking less than 2 minutes to train on a single GPU, since they are O(1) in sequence length after obtaining the mean sequence embeddings. This indicates that the pretrained models have learned rich embeddings of sequence and structure.

Additionally, the finetuned structure model achieves a substantially better AUPRC (0.566 vs 0.446) and max F1 score (0.575 vs 0.460) than DeepFRI. The gap between our structural and non-structural models is much wider than that of DeepFRI, indicating that our model derives a greater benefit from structural information, despite the fact that DeepFRI uses established models for structural processing.

Table 1: GO molecular function prediction results.

Pretraining (# seqs)	Method	Structure	AUPRC	(Gain from structure)	Max F1	(Gain from structure)
DeepFRI (∼10M)	DeepFRI	✗	0.427		0.438	
		✓	0.446	0.019	0.460	0.022
None	DeepCNN	✗	0.363		0.385	
Ours (∼35K)	MLP	✗	0.361		0.377	
		✓	0.460	0.099	0.465	0.088
	Finetuned	✗	0.381		0.421	
		✓	0.566	0.185	0.575	0.154

4 CONCLUSIONS

In this work we show that standard Transformers are capable of performing structural reasoning by learning an approximately SE(3)-invariant distance filter on attention. We predict that even linearly embedded positions can produce Gaussian attention filters of distance and validate this prediction using experiments on simulated points and proteins. The protein model naturally learns to use the 3D coordinates to measure distance which substantially improves its ability to predict masked tokens. The structural information also materially improves the model's ability to inform function prediction, providing even greater benefit than existing custom-built structural models.

We show that Transformers can learn to measure distance and operate as hybrid structure/language models. In contrast to many conventional structure models which are based on GNNs, Transformers do not explicitly model edges. This admits memory-efficient implementations such as FlashAttention (Dao et al., 2022; Dao, 2023) which allow for fast, fully-connected updates in linear memory. Most structure models store distance in edges which use quadratic memory for fully-connected graphs. Practically, this means that Transformers can perform structural reasoning on more highly connected structures, which may allow them to "see" more while making decisions.

As shown in Section 3.2, the pretrained protein model trained with coordinates showed a strong positional dependence in attention in early layers followed by a weak positional dependence in the last few layers. It is possible that this corresponds to the model identifying structural features such as secondary structure and local physics, before encoding these and performing long-range sequential processing. This corresponds with the contemporary trend of preprocessing structural information to create structural tokens for Transformers compared to the more traditional approach of using language model embeddings as input to structural GNNs.

In this work we explore two protein tasks: masked token prediction and function prediction. Virtually all protein learning tasks benefit from combined sequence and structure processing and so this work could be applied across areas including inverse folding, structure prediction, and arbitrary property prediction. As is common in tasks such as inverse folding, the input structures could include more atoms from the backbone. This could be achieved by simply projecting these atom coordinates to the input representation, unlike GNN-based methods which require explicitly including all pairwise distances in the edge features. Additionally, while proteins are a natural fit for structural Transformers due to their combined sequential and spatial data, there are many other possible applications of this type of model. Some of these include tasks with explicit 3D information such as small molecules and 3D objects. However, there are also tasks where learning an approximate relationship between entities in Euclidean space could help with reasoning, such as vision Transformers (Dosovitskiy et al., 2021) or even large language models.

MEANINGFULNESS STATEMENT

Proteins, the molecular machines of life, are typically represented as either sequences or structures. These two representations are often used with different types of models: Transformers can leverage long-range sequential patterns, while GNNs can interpret local structural features. There has been interest in combining these representations by using sequence-level embeddings as node annotations for GNNs or using GNNs to create structural tokens for Transformers. In this work, we show that Transformers are capable of creating their own meaningful representation of protein coordinates, which allows them to learn to attend to local structure and operate as a hybrid sequence-structure model.

ACKNOWLEDGMENTS

IE was supported by funding from the Engineering and Physical Sciences Research council [EP/S024093/1] and Exscientia.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, May 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-07487-w. URL https://www.nature.com/articles/s41586-024-07487-w.

Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000. ISSN 1546-1718. doi: 10.1038/75556. URL https://www.nature.com/articles/ng0500_25. Publisher: Nature Publishing Group.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. URL http://arxiv.org/abs/1607.06450. arXiv:1607.06450 [cs, stat].

David A. C. Beck, Darwin O. V. Alonso, Daigo Inoyama, and Valerie Daggett. The intrinsic conformational propensities of the 20 naturally occurring amino acids and reflection of these propensities in proteins. *Proceedings of the National Academy of Sciences*, 105(34):12259–12264, August 2008. doi: 10.1073/pnas.0706527105. URL https://www.pnas.org/doi/full/10.1073/pnas.0706527105. Publisher: Proceedings of the National Academy of Sciences.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning, July 2023. URL http://arxiv.org/abs/2307.08691. arXiv:2307.08691 [cs].

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, June 2022. URL http://arxiv.org/abs/2205.14135. arXiv:2205.14135 [cs].

Nima Dehmamy, Albert-Laszlo Barabasi, and Rose Yu. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/73bf6c41e241e28b89d0fb9e0c82f9ce-Abstract.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv:1810.04805 [cs].

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs].

Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks, November 2020. URL http://arxiv.org/abs/2006.10503. arXiv:2006.10503 [cs, stat].

Vladimir Gligorijević, P. Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C. Taylor, Ian M. Fisk, Hera Vlamakis, Ramnik J. Xavier, Rob Knight, Kyunghyun Cho, and Richard Bonneau. Structure-based protein function prediction using graph convolutional networks. *Nature Communications*, 12(1):3168, May 2021.

ISSN 2041-1723. doi: 10.1038/s41467-021-23303-9. URL https://www.nature.com/articles/s41467-021-23303-9. Publisher: Nature Publishing Group.

Alex Graves. Generating Sequences With Recurrent Neural Networks, June 2014. URL http://arxiv.org/abs/1308.0850. arXiv:1308.0850 [cs].

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model, July 2024. URL https://www.biorxiv.org/content/10.1101/2024.07.01.600583v1. Pages: 2024.07.01.600583 Section: New Results.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), June 2023. URL http://arxiv.org/abs/1606.08415. arXiv:1606.08415 [cs] version: 5.

Bosco K. Ho and Robert Brasseur. The Ramachandran plots of glycine and pre-proline. *BMC Structural Biology*, 5(1):14, August 2005. ISSN 1472-6807. doi: 10.1186/1472-6807-5-14. URL https://doi.org/10.1186/1472-6807-5-14.

Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. preprint, Systems Biology, April 2022. URL http://biorxiv.org/lookup/doi/10.1101/2022.04.10.487779.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative Models for Graph-Based Protein Design. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/f3a4ff4839c56a5f460c88cce3666a2b-Abstract.html.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2. Number: 7873 Publisher: Nature Publishing Group.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. July 2022. URL https://openreview.net/forum?id=SJU4ayYgl.

Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, February 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx624. URL https://doi.org/10.1093/bioinformatics/btx624.

Maxat Kulmanov, Francisco J. Guzmán-Vega, Paula Duek Roggli, Lydie Lane, Stefan T. Arold, and Robert Hoehndorf. Protein function prediction as approximate semantic entailment. *Nature Machine Intelligence*, 6(2):220–228, February 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00795-w. URL https://www.nature.com/articles/s42256-024-00795-w. Publisher: Nature Publishing Group.

Yann LeCun, Yoshua Bengio, and others. Convolutional networks for images, speech, and time series. Publisher: Citeseer.

Mingchen Li, Pan Tan, Xinzhu Ma, Bozitao Zhong, Huiqun Yu, Ziyi Zhou, Wanli Ouyang, Bingxin Zhou, Liang Hong, and Yang Tan. ProSST: Protein Language Modeling with Quantized Structure and Disentangled Attention, May 2024. URL https://www.biorxiv.org/content/10.1101/2024.04.15.589672v3. Pages: 2024.04.15.589672 Section: New Results.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs, February 2023. URL http://arxiv.org/abs/2206.11990. arXiv:2206.11990 [physics].

Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. EquiformerV2: Improved Equivariant Transformer for Scaling to Higher-Degree Representations, December 2023. URL http://arxiv.org/abs/2306.12059. arXiv:2306.12059 [physics].

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. doi: 10.1126/science.ade2574. URL https://www.science.org/doi/10.1126/science.ade2574. Publisher: American Association for the Advancement of Science.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, April 2021. doi: 10.1073/pnas.2016239118. URL https://www.pnas.org/doi/10.1073/pnas.2016239118. Publisher: Proceedings of the National Academy of Sciences.

Noam Shazeer. GLU Variants Improve Transformer, February 2020. URL http://arxiv.org/abs/2002.05202. arXiv:2002.05202 [cs, stat].

Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, November 2017. ISSN 1546-1696. doi: 10.1038/nbt.3988. URL https://www.nature.com/articles/nbt.3988. Publisher: Nature Publishing Group.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding, August 2022. URL http://arxiv.org/abs/2104.09864. arXiv:2104.09864 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. URL http://arxiv.org/abs/1706.03762. arXiv:1706.03762 [cs].

Jesse Vig, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Rajani. BERTology Meets Biology: Interpreting Attention in Protein Language Models. October 2020. URL https://openreview.net/forum?id=YWtLZvLmud7.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On Layer Normalization in the Transformer Architecture. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10524–10533. PMLR, November 2020. URL https://proceedings.mlr.press/v119/xiong20b.html. ISSN: 2640-3498.

A Appendix

A.1 Prior work

There have been prior approaches to merge Transformers with SE(3)-(in/equi)variant models, especially for computational chemistry and 3D point clouds. Some methods add attention blocks to SE(3) GNNs to create SE(3)-invariant GNN Transformers (Fuchs et al., 2020; Liao & Smidt, 2023). These have shown good results on a number of tasks, however tend to be memory-intensive, particularly because attention is performed on edges, which grow as n^2 for fully-connected graphs. As a result, the graph connectedness of the GNNs is typically limited to k-nearest neighbours. In contrast, memory-efficient attention implementations such as FlashAttention (Dao et al., 2022; Dao, 2023) have enabled linear-memory standard Transformers.

Previous works have demonstrated that sequence-only protein Transformers can learn attention maps which correlate with physical contacts (Lin et al., 2023; Vig et al., 2020). However, these works do not formally model structure and so are limited by which contacts can be predicted from purely sequential patterns. To overcome this, Transformers are often paired with structural models, for instance methods such as ProSST (Li et al., 2024), ESM-IF (Hsu et al., 2022), and ESM3 (Hayes et al., 2024) use custom graph-based modules to create structural tokens which are fed into standard Transformers. In contrast, our work shows that standard Transformers are natively capable of using coordinates to model structure and measure distance.

Similarly, AlphaFold2 (Jumper et al., 2021) and ESMFold (Lin et al., 2023) use Transformers to preprocess protein sequences for structure prediction. Again, these preprocessed representations have been shown to correlate with structural contacts. AlphaFold2 makes this explicit during training by minimizing a distrogram loss which encourages the EvoFormer to learn structural contacts. However, it is unclear if these representations are learning to explicitly embed coordinates in 3D and both models still require an SE(3)-equivariant GNN structure module to actually produce 3D structures.

In building AlphaFold3, DeepMind replaced AlphaFold2's SE(3)-equivariant structure module with linearly embedded coordinates fed into a diffusion transformer (Abramson et al., 2024). AlphaFold3's structure module uses inner product attention with a pair bias learned from the pair representation. At present, this still requires quadratic memory, however does indicate that nearly-standard Transformers with linearly embedded coordinates can learn on structure. Here, we explore how such linearly embedded coordinates can be used by standard Transformer attention modules to measure the Euclidean distance between tokens, and, in contrast to prior work, show that no modifications are necessary for the standard Transformer architecture to learn to perform structural reasoning.

A.2 Coordinates can be rescaled for better approximations

The validity of the approximations shown so far depends on the coordinates being small. Figure A1 shows how well $(cLN(E_{lin}(\frac{x_1}{c})))\cdot cLN(E_{lin}(\frac{x_2}{c}))$ approximates a quadratic as a function of c and how well the resulting exponential approximates a Gaussian. The scaling parameter c can be learned by the input and output linear maps of the embedding or by the LayerNorm gain parameters. In this way, all coordinates can be rescaled such that the previous sections produce arbitrarily good approximations.

A.3 Embedding proofs

A.3.1 Trigonometric embeddings

Consider the embedding E_{trig}:

$$E_{trig}(x) = (\cos(x), -\cos(x), \sin(x), -\sin(x)) \tag{A1}$$

Then the mean, $\mu(E_{trig}(x))$, is:

$$\mu(E_{trig}(x)) = 0 \tag{A2}$$

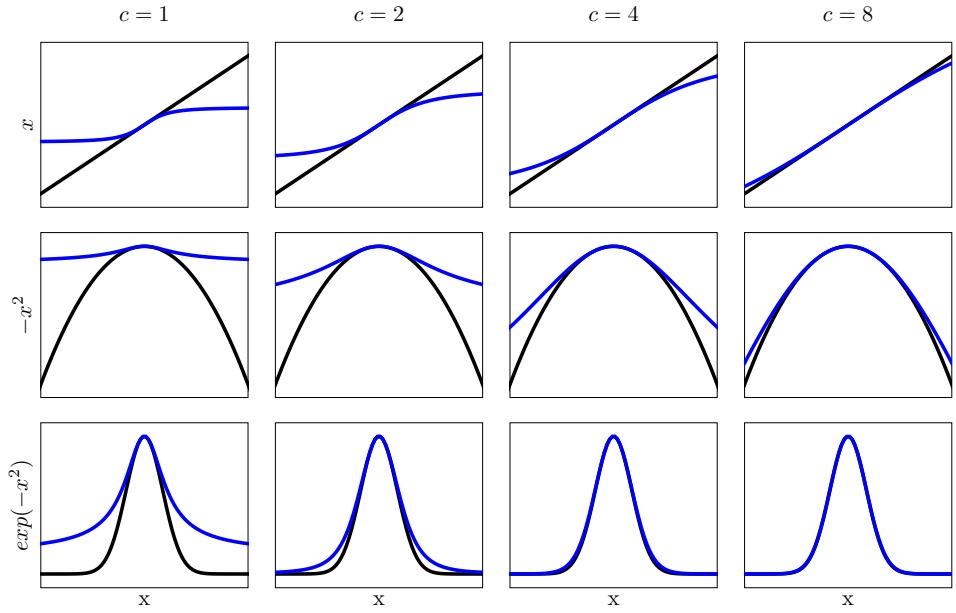


Figure A1: Linear and quadratic approximation and resulting Gaussian approximation of $cLN(E_{lin}(\frac{x}{c}))$ as a function of c. Target functions are shown in black and approximations are shown in blue. Increasing the scaling parameter c results in a better approximation without changing the shape of the underlying Gaussian.

and the variance, $\sigma(E_{trig}(x))$, is:

$$\begin{aligned}\sigma(E_{trig}(x)) &= \sqrt{\frac{1}{4}(\sin(x)^2 + (-\sin(x))^2 + \cos(x)^2 + (-\cos(x))^2)} \\ &= \sqrt{\frac{1}{2}(\sin(x)^2 + \cos(x)^2)} \\ &= \frac{1}{\sqrt{2}}\end{aligned} \tag{A3}$$

So

$$LN(E_{trig}(x)) = \frac{E_{trig}(x) - \mu}{\sigma} = \frac{E_{trig}(x) - 0}{\frac{1}{\sqrt{2}}} = \sqrt{2}E_{trig}(x) \tag{A4}$$

$$\begin{aligned}LN(E_{trig}(x_1)) \cdot LN(E_{trig}(x_2)) &= \sqrt{2}^2(2\cos(x_1)\cos(x_2) + 2\sin(x_1)\sin(x_2)) \\ &= 4(\cos(x_1 - x_2)) \\ &\approx 4 - 2(x_1 - x_2)^2\end{aligned} \tag{A5}$$

A.3.2 Layer normalization can learn approximately quadratic functions of input

Consider the first-order approximation of E_{trig}, E_{lin}:

$$E_{lin}(x) = (1, -1, x, -x) \tag{A6}$$

We have

$$\mu(E_{lin}(x)) = 1 - 1 + x - x = 0 \tag{A7}$$

$$\begin{aligned}
\sigma(E_{lin}(x)) &= \sqrt{\frac{1}{4}(1^2 + (-1)^2 + x^2 + (-x)^2)} \\
&= \sqrt{\frac{1}{2}(1 + x^2)} \\
&\approx \sqrt{\frac{1}{2}(1 + x^2 + \frac{x^4}{4})} \\
&= \sqrt{\frac{1}{2}(1 + \frac{x^2}{2})^2} \\
&= \frac{1}{\sqrt{2}}(1 + \frac{x^2}{2})
\end{aligned} \tag{A8}$$

$$\begin{aligned}
\frac{1-\mu}{\sigma} &= \frac{1}{\frac{1}{\sqrt{2}}(1 + \frac{x^2}{2})} \\
&= \sqrt{2}\frac{1 - \frac{x^2}{2}}{(1 + \frac{x^2}{2})(1 - \frac{x^2}{2})} \\
&= \sqrt{2}\frac{1 - \frac{x^2}{2}}{1 - \frac{x^4}{4}} \\
&\approx \sqrt{2}(1 - \frac{x^2}{2})
\end{aligned} \tag{A9}$$

$$\begin{aligned}
\frac{x-\mu}{\sigma} &= \frac{x}{\frac{1}{\sqrt{2}}(1 + \frac{x^2}{2})} \\
&\approx \sqrt{2}(x)
\end{aligned} \tag{A10}$$

So,

$$LN((1, -1, x, -x)) \approx \sqrt{2}((1 - \frac{x^2}{2}), -(1 - \frac{x^2}{2}), x, -x) \tag{A11}$$

In this way, layer normalization can be used to generate approximately quadratic functions of the input. In particular,

$$\begin{aligned}
LN(E_{lin}(x_1)) \cdot LN(E_{lin}(x_2)) &\approx \sqrt{2}^2(2(1 - \frac{x_1^2}{2})(1 - \frac{x_2^2}{2}) + 2(x_1 x_2)) \\
&= 4(1 - \frac{x_1^2}{2} - \frac{x_2^2}{2} + \frac{x_1^2 x_2^2}{4} + x_1 x_2) \\
&= 4(\frac{1}{2}(2 - (x_1^2 - 2x_1 x_2 + x_2^2) + \frac{x_1^2 x_2^2}{2})) \\
&= 2(-(x_1 - x_2)^2 + 2 + \frac{x_1^2 x_2^2}{2}) \\
&\approx -2(x_1 - x_2)^2 + 4
\end{aligned} \tag{A12}$$

A.3.3 GATED LINEAR UNITS PROVIDE A BETTER APPROXIMATION

Lemma A.1. *ReGLU and SwiGLU can produce functions of x^2. In particular:*

$$ReGLU(x) + ReGLU(-x) = SwiGLU(x) + SwiGLU(-x) = x^2$$

Proof.

$$\begin{aligned} ReGLU(x) + ReGLU(-x) &= max(0, x)x + max(0, -x)x \\ &= max(-x, x)x \\ &= |x|x \\ &= x^2 \end{aligned} \tag{A13}$$

Similarly,

$$\begin{aligned} SwiGLU(x) + SwiGLU(-x) &= \frac{x^2}{1 + e^{-x}} + \frac{(-x)^2}{1 + e^{-(-x)}} \\ &= \frac{x^2(1 + e^x)}{(1 + e^{-x})(1 + e^x)} + \frac{x^2(1 + e^{-x})}{(1 + e^x)(1 + e^{-x})} \\ &= \frac{x^2(1 + e^x) + x^2(1 + e^{-x})}{1 + e^{-x} + e^x + e^{x-x}} \\ &= \frac{x^2(2 + e^x + e^{-x})}{2 + e^x + e^{-x}} \\ &= x^2 \end{aligned} \tag{A14}$$

□

Theorem A.2. *With input $\vec{x} = (1, x, x^2)$, there exists a linear embedding, E, and a linear map L such that $L(LN(E(\vec{x}))) = (1, x, x^2)/\sigma$ where $1 \leq \sigma \leq 1 + \frac{x^4}{8}$*

Proof. Consider the second-order approximation of E_{trig}, $E_{quad}(x)$:

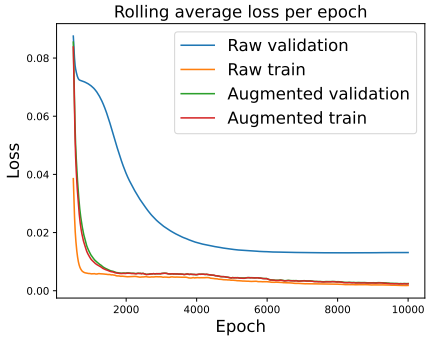
$$E_{quad}(x) = (1 - \frac{x^2}{2}, -(1 - \frac{x^2}{2}), x, -x) \tag{A15}$$

We have

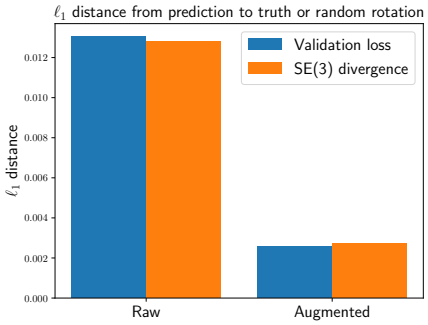
$$\mu(E_{quad}(x)) = 0 \tag{A16}$$

$$\begin{aligned} \sigma(E_{quad}(x)) &= \sqrt{\frac{1}{4}(x^2 + (-x)^2 + (1 - \frac{x^2}{2})^2 + (-(1 - \frac{x^2}{2}))^2)} \\ &= \sqrt{\frac{1}{4}(2 + \frac{2x^4}{4})} \\ &= \frac{1}{\sqrt{2}}\sqrt{1 + \frac{x^4}{4}} \\ &\approx \frac{1}{\sqrt{2}}\sqrt{1 + \frac{x^4}{4} + \frac{x^8}{64}} \\ &= \frac{1}{\sqrt{2}}\sqrt{(1 + \frac{1}{8}x^4)^2} \\ &= \frac{1}{\sqrt{2}}(1 + \frac{1}{8}x^4) \end{aligned} \tag{A17}$$

□



(a) Loss vs epoch



(b) Loss and SE(3) divergence

Figure A2: (a) Training and validation loss per epoch in a low data setting. The rolling average of the last 500 epochs is shown. Raw coordinates show large divergence between training and validation loss whereas randomly rotated coordinates show near perfect alignment, resulting in a substantially lower validation loss. (b) The final validation loss is very close to the average predicted distance between randomly rotated structures, indicating that the validation loss is minimized by learning a more SE(3)-invariant measure of distance.

The error for the LayerNorm-only approximation of x and x^2 was $O(n^3)$ and $O(n^4)$ respectively. In comparison, the error given x^2 as input for x and x^2 is $O(n^5)$ and $O(n^6)$. Thus, simple combinations of ReGLU or SwiGLU layers give us a better approximation of x and x^2, which in turn gives us a better approximation of d^2. In practice, this may mean that x need not be as small for reasonable approximations to hold which may allow for more stable gradients.

A.4 SE(3) TRANSFORMATIONS IMPROVE LEARNED SE(3)-INVARIANCE

We investigated whether Transformers will learn to overfit training data in a low data regime and if this can be prevented. This could also correspond to a scenario where there is only a strong structural signal in a small number of training examples. We reduced the number of training points to 100 and measured the training and validation loss. To test the importance of data augmentation, we trained models with and without the random rotations (Figure A2a). The raw coordinates clearly demonstrate overfitting while the randomly rotated coordinates show near perfect alignment between training and validation loss. Importantly, this form of data augmentation does not require creating new data points, only rotating the training data each epoch.

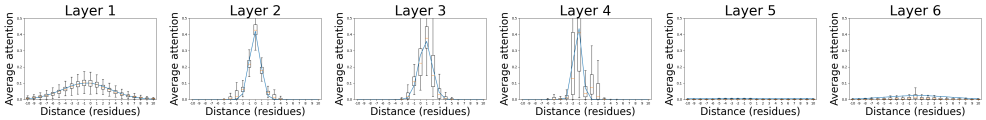
We measured the average ℓ_1 distance between predictions of randomly rotated structures for the models trained with and without random rotations, as a measure of SE(3) divergence (Figure A2b). In both cases the SE(3) divergence was almost the same as the validation loss, indicating that randomly rotating training structures reduces overfitting by encouraging models to learn an SE(3)-invariant measure of distance.

A.5 RAW ATTENTION PLOTS

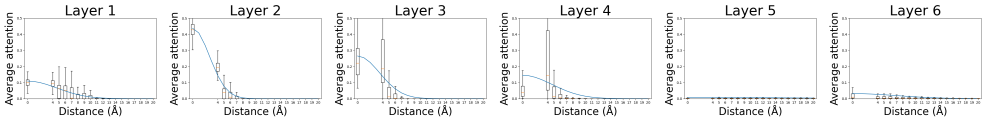
In Section 3.2, we showed that the attention paid to positional and 3D distance are well-fit by Gaussians. Here, in Figure A3, we provide the same plots but without isolating each of the factors. Note that relative distance, position, and amino acid type are all correlated with one another, especially at linear/3D distance 0.

A.6 EXTENDED EXPERIMENTS

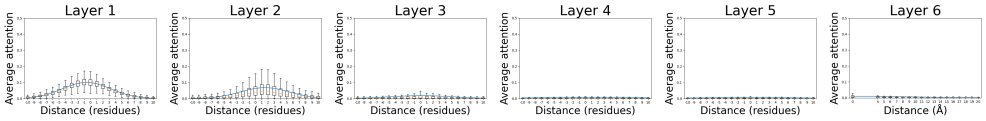
Here, we report the results of two additional experiments on predicting biological process (Table 2) and molecular function (Table 3) labels, also from the DeepFRI dataset. We were unable to compare these results to DeepFRI because the PDB-only results were not reported. As in the molecular



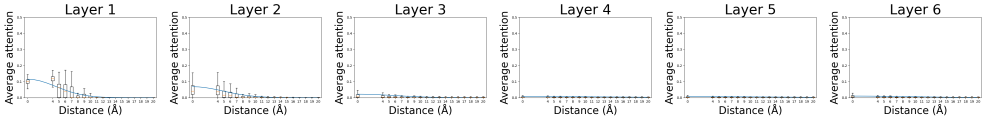
(a) Linear position, with coordinates



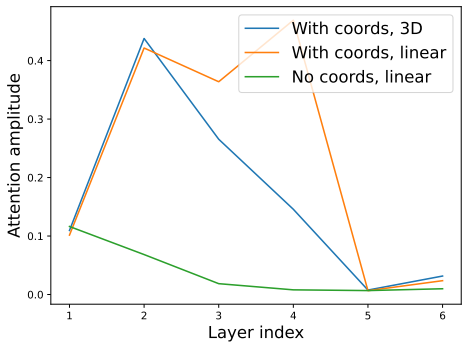
(b) 3D position, with coordinates



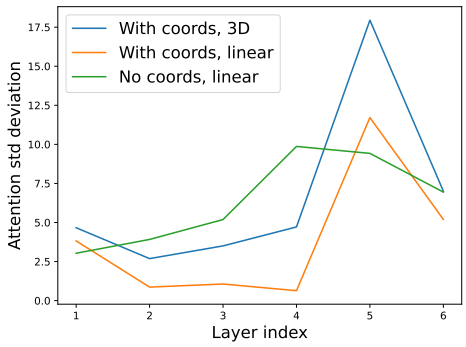
(c) Linear position, no coordinates



(d) 3D position, no coordinates



(e) Gaussian amplitudes



(f) Gaussian standard deviations

Figure A3: Average attention paid per layer to linear and 3D positional information (a-d) for the unmodified inputs. Fit Gaussian functions are shown in blue for each plot. Amplitudes (e) and standard deviations (f) of the fit Gaussians are shown. The amplitudes are higher and the fits are worse due to the cross-correlation between relative position, 3D distance, and amino acid type.

function experiments, the inclusion of structure improved the performance of all models. In these experiments, the performance of the MLP and finetuned Transformers were comparable, and the sequence-only MLP outperformed the sequence-only Transformer for cellular component prediction. It is possible that this is the result of the more expressive Transformer model overfitting to sequence training data, which is then mitigated by the inclusion of structure.

Table 2: GO biological process prediction results.

Pretraining (# seqs)	Method	Structure	AUPRC	(Gain from structure)	Max F1	(Gain from structure)
Ours (∼35K)	MLP	✗	0.197		0.247	
		✓	0.235	0.038	0.286	0.039
	Finetuned	✗	0.191		0.232	
		✓	0.244	0.053	0.280	0.048

Table 3: GO cellular component prediction results.

Pretraining (# seqs)	Method	Structure	AUPRC	(Gain from structure)	Max F1	(Gain from structure)
Ours (∼35K)	MLP	✗	0.281		0.335	
		✓	0.306	0.025	0.350	0.015
	Finetuned	✗	0.230		0.271	
		✓	0.308	0.078	0.343	0.062

A.7 Model parameter counts

In Table 4, we list the parameter counts for all models used in the paper. The simulated parameter counts will vary slightly as described in the details of each experiment. The pretrained models have the same number of parameters both with and without coordinates. The finetuned models have a slightly higher number of parameters than the pretrained because of the final linear layer which projects to the number of classes. The MLP parameter counts are relatively low because they are conditioned on the (fixed) pretrained embeddings.

Table 4: Parameter counts for all models.

Model name	count
Simulated model	1,597,504
Pretrained models	33,129,242
Finetuned models (cc)	33,375,322
Finetuned models (mf)	33,505,283
Finetuned models (bp)	34,623,409
MLP models (cc)	2,169,152
MLP models (mf)	2,342,377
MLP models (bp)	3,832,727