LEARNING FROM PROTEIN STRUCTURE WITH GEOMETRIC VECTOR PERCEPTRONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning on 3D structures of large biomolecules is emerging as a distinct area in machine learning, but there has yet to emerge a unifying network architecture that simultaneously leverages the geometric and relational aspects of the problem domain. To address this gap, we introduce *geometric vector perceptrons*, which extend standard dense layers to operate on collections of Euclidean vectors. Graph neural networks equipped with such layers are able to perform both geometric and relational reasoning on efficient representations of macromolecules. We demonstrate our approach on two important problems in learning from protein structure: model quality assessment and computational protein design. Our approach improves over existing classes of architectures on both problems, including state-of-the-art convolutional neural networks and graph neural networks.

1 INTRODUCTION

Many efforts in structural biology aim to predict, or derive insights from, the structure of a macromolecule (such as a protein, RNA, or DNA), represented as a set of positions associated with atoms or groups of atoms in 3D Euclidean space. These problems can often be framed as functions mapping the input domain of structures to some property of interest—for example, predicting the quality of a structural model or determining whether two molecules will bind. Thanks to their importance and difficulty, such problems, which we broadly refer to as *learning from structure*, have recently developed into an exciting and promising application area for deep learning (Graves et al., 2020; Ingraham et al., 2019; Pereira et al., 2016; Townshend et al., 2019; Won et al., 2019).

Successful applications of deep learning are often driven by techniques that leverage the problem structure of the domain—for example, convolutions in computer vision (Cohen & Shashua, 2016) and attention in natural language processing (Vaswani et al., 2017). What are the relevant considerations in the domain of learning from structure? Using proteins as the most common example, we have on the one hand the arrangement and orientation of the amino acids in space, which govern the dynamics and function of the molecule (Berg et al., 2002). On the other hand, proteins also possess relational structure in terms of their amino-acid sequence and the residue-residue interactions that mediate the aforementioned protein properties (Hammes-Schiffer & Benkovic, 2006). We refer to these as the *geometric* and *relational* aspects of the problem domain, respectively.

Recent state-of-the-art methods for learning from structure are successful by leveraging one of these two aspects. Commonly, such methods either employ graph neural networks (GNNs), which are expressive in terms of relational reasoning (Battaglia et al., 2018), or convolutional neural networks (CNNs), which operate directly on the geometry of the structure. Here, we present a unifying architecture that bridges these two families of methods to leverage *both* aspects of the problem domain.

We do so by introducing *geometric vector perceptrons* (GVPs), a drop-in replacement for standard multi-layer perceptrons (MLPs) in aggregation and feed-forward layers of GNNs. GVPs operate directly on both scalar and *geometric* features—features that transform as a vector under a rotation of spatial coordinates. GVPs therefore allow for the embedding of geometric information at nodes and edges without reducing such information to scalars that may not fully capture complex geometry. We postulate that our approach makes it easier for a GNN to learn functions whose significant features are both geometric and relational.

Our method (GVP-GNN) can be applied to any problem where the input domain is a structure of a single macromolecule or of molecules bound to one another. In this work, we specifically demonstrate our approach on two problems connected to protein structure: model quality assessment and computational protein design. Model quality assessment (MQA) aims to select the best structural model of a protein from a large pool of candidate structures and is a crucial step in protein structure prediction (Cheng et al., 2019). Computational protein design (CPD) is the conceptual inverse of structure prediction, aiming to infer an amino acid sequence that will fold into a given structure. Our method outperforms existing methods on both tasks.

2 Related Work

ML methods for learning from protein structure largely fall into one of three types, operating on *sequential*, *voxelized*, or *graph-structured* representations of proteins. We briefly discuss each type and introduce state-of-the-art examples for MQA and CPD to set the stage for our experiments later.

Sequential representations In traditional models of learning from protein structure, each amino acid is represented as a feature vector using hand-crafted representations of the 3D structural environment, such as contact-based features (Olechnovič & Venclovas, 2017), orientations or positions collectively projected to local coordinates (Karasikov et al., 2019; Wang et al., 2018), and physics-inspired energy terms (O'Connell et al., 2018; Uziela et al., 2017). The structure is then viewed as a sequence of such feature vectors which can be fed into a 1D convolutional network, RNN, or dense feedforward network. Although these methods only indirectly represent the full 3D structure of the protein, a number of them, such as ProQ3D (Uziela et al., 2017), VoroMQA (Olechnovič & Venclovas, 2017), and SBROD (Karasikov et al., 2019), are competitive in assessments of MQA.

Voxelized representations In lieu of hand-crafted representations of structure, 3D convolutional neural networks (CNNs) can operate directly on the positions of atoms in space, encoded as occupancy maps in a voxelized 3D volume. The hierarchical convolutions of such networks are easily compatible with the detection of structural motifs, binding pockets, and the specific shapes of other important structural features, leveraging what we call the *geometric* aspect of the domain. The MQA methods 3DCNN (Derevyanko et al., 2018) and Ornate (Pagès et al., 2019) and a number of CPD methods (Anand et al., 2020; Zhang et al., 2019) exemplify the power of this approach.

Graph-structured representations A 3D protein structure can also be represented as a proximity graph over amino acid nodes, reducing the challenging task of representing a collective structural neighborhood in a single feature vector to that of representing individual edges. Graph neural networks (GNNs) can then perform complex *relational* reasoning over structures (Battaglia et al., 2018)—perhaps identifying key relationships among a set of amino acids, or flexible structural motifs best described as a connectivity pattern rather than a rigid shape. While some GNN methods, such as GraphQA (MQA) (Baldassarre et al., 2020), simply represent edges by their length, others, such as Structured Transformer (CPD) (Ingraham et al., 2019), indirectly encode the 3D geometry of the proximity graph in terms of relative orientations and other scalar features.

3 Methods

The GNNs described in the previous section encode the 3D geometry of the protein by encoding *vector* features (such as node orientations and edge directions) in terms of rotation-invariant *scalars*, often by defining a local coordinate system at each node. We instead propose that these features be *directly represented as geometric vectors*—features in $\mathbb{R}^3$ which transform appropriately under a change of spatial coordinates—at all steps of graph propagation. This conceptual shift has two important ramifications. First, the input representation is more efficient: instead of encoding the orientation of a node by its relative orientation with all of its neighbors, we only have to represent one absolute orientation per node. Second, it standardizes a global coordinate system across the entire structure, which allows geometric features to be directly propagated without transforming between local coordinates. We postulate this allows the GNN to more easily access global geometric properties of the structure. The key challenge with such a representation, however, is to perform graph propagation in a way that simultaneously preserves the full expressive power of the original

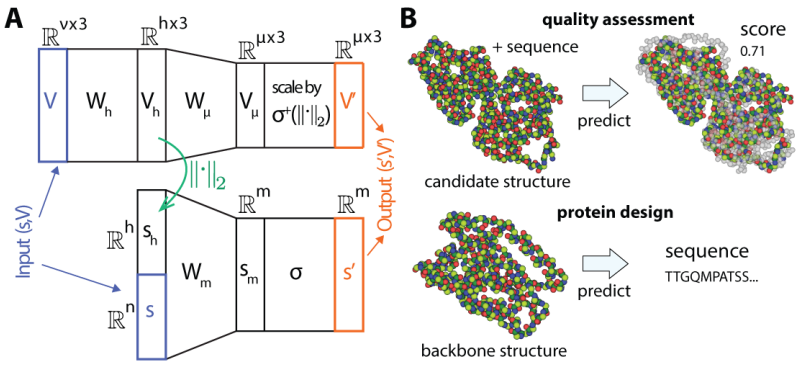

Figure 1: **(A)** Schematic of the geometric vector perceptron illustrating Algorithm 1. Given a tuple of scalar and vector input features $(\mathbf{s}, \mathbf{V})$, the perceptron computes an updated tuple $(\mathbf{s}', \mathbf{V}')$. $\mathbf{s}'$ is a function of both $\mathbf{s}$ and $\mathbf{V}$. **(B)** Illustration of the structure-based prediction tasks. In model quality assessment (top), the goal is to predict a quality score given the 3D structure of a candidate model. Individual atoms are represented as colored spheres. The quality score measures the accuracy of a candidate structure with respect to an experimentally determined structure (shown in gray). In computational protein design (bottom), the goal is to predict an amino acid sequence that would fold into a given protein backbone structure.

GNN while maintaining the rotation invariance provided by the scalar representations. We do so by introducing a new module, the *geometric vector perceptron*, to replace dense layers in a GNN.

3.1 Geometric Vector Perceptrons

The geometric vector perceptron is a simple module for learning vector-valued and scalar-valued functions over geometric vectors and scalars. That is, given a tuple $(\mathbf{s}, \mathbf{V})$ of scalar features $\mathbf{s} \in \mathbb{R}^n$ and vector features $\mathbf{V} \in \mathbb{R}^{\nu \times 3}$, we compute new features $(\mathbf{s}', \mathbf{V}') \in \mathbb{R}^m \times \mathbb{R}^{\mu \times 3}$. The computation is formally described in Algorithm 1 and illustrated in Figure 1A.

At its core, the GVP consists of two separate linear transformations $\mathbf{W}_m, \mathbf{W}_h$ for the scalar and vector features, followed by nonlinearities σ, σ^+. However, before the scalar features are transformed, we concatenate the L_2 norm of the transformed vector features $\mathbf{V}_h$; this allows us to extract rotation-invariant information from the input vectors $\mathbf{V}$. An additional linear transformation $\mathbf{W}_\mu$ is inserted just before the vector nonlinearity to control the output dimensionality independently of the number of norms extracted.

Algorithm 1 Geometric vector perceptron

Input: Scalar and vector features $(\mathbf{s}, \mathbf{V}) \in \mathbb{R}^n \times \mathbb{R}^{\nu \times 3}$.
Output: Scalar and vector features $(\mathbf{s}', \mathbf{V}') \in \mathbb{R}^m \times \mathbb{R}^{\mu \times 3}$.
$h \leftarrow \max(\nu, \mu)$
GVP:
$\mathrm{V}_h \leftarrow \mathbf{W}_h \mathbf{V} \quad \in \mathbb{R}^{h \times 3}$
$\mathrm{V}_\mu \leftarrow \mathbf{W}_\mu \mathrm{V}_h \quad \in \mathbb{R}^{\mu \times 3}$
$\mathrm{s}_h \leftarrow \|\mathrm{V}_h\|_2$ (row-wise) $\quad \in \mathbb{R}^h$
$\mathrm{v}_\mu \leftarrow \|\mathrm{V}_\mu\|_2$ (row-wise) $\quad \in \mathbb{R}^\mu$
$\mathrm{s}_{h+n} \leftarrow \mathrm{concat}(\mathrm{s}_h, \mathbf{s}) \quad \in \mathbb{R}^{h+n}$
$\mathrm{s}_m \leftarrow \mathbf{W}_m \mathrm{s}_{h+n} + \mathbf{b} \quad \in \mathbb{R}^m$
$\mathbf{s}' \leftarrow \sigma(\mathrm{s}_m) \quad \in \mathbb{R}^m$
$\mathbf{V}' \leftarrow \sigma^+(\mathrm{v}_\mu) \odot \mathrm{V}_\mu$, (row-wise multiplication) $\quad \in \mathbb{R}^{\mu \times 3}$
return $(\mathbf{s}', \mathbf{V}')$

The GVP is conceptually simple, yet provably possesses the desired properties of invariance/equivariance and expressiveness. First, the vector and scalar outputs of the GVP are equivariant and invariant, respectively, with respect to an arbitrary composition of rotations and reflections in

3D Euclidean space described by R *i.e.*,

$$\text{GVP}((\mathbf{s}, R(\mathbf{V}))) = (\mathbf{s}', R(\mathbf{V}')) \tag{1}$$

This is due to the fact that the only operations on vector-valued inputs are scalar multiplication, linear combination, and the L_2 norm.[1] We include a formal proof in Appendix A.

In addition, a GVP can approximate *any* continuous rotation- and reflection-invariant scalar-valued function of $\mathbf{V}$. More precisely, let G_s be a GVP defined with $n, \mu = 0$—that is, the part of a GVP that transforms vector features to scalar features. Then G_s is able to ϵ-approximate a function $F : \mathbb{R}^{\nu \times 3} \to \mathbb{R}$ that is invariant with respect to rotations and reflections in 3D under mild assumptions.

Theorem. *Let R describe an arbitrary rotation or reflection in $\mathbb{R}^3$. For $\nu \geq 3$ let $\Omega^\nu \subset \mathbb{R}^{\nu \times 3}$ be the set of all $\mathbf{V} = [\mathbf{v}_1, \quad \ldots, \quad \mathbf{v}_\nu]^T \in \mathbb{R}^{\nu \times 3}$ such that $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are linearly independent and $0 < ||\mathbf{v}_i||_2 \leq b$ for all i and some finite $b > 0$. Then for any continuous $F : \Omega^\nu \to \mathbb{R}$ such that $F(R(\mathbf{V})) = F(\mathbf{V})$ and for any $\epsilon > 0$, there exists a form $f(\mathbf{V}) = \mathbf{w}^T G_s(\mathbf{V})$ such that $|F(\mathbf{V}) - f(\mathbf{V})| < \epsilon$ for all $\mathbf{V} \in \Omega^\nu$.*

We include a formal proof in Appendix A. As a corollary, a GVP with nonzero n, μ is also able to approximate similarly-defined functions over the full input domain $\mathbb{R}^n \times \mathbb{R}^{\nu \times 3}$.

In addition to the GVP layer itself, we use a version of dropout that drops entire vector channels at random (as opposed to coordinates within vector channels). We also introduce layer normalization for the vector features as

$$\mathbf{v}_i \leftarrow \mathbf{v}_i / \sqrt{\frac{1}{|\{j\}|} \sum_j ||\mathbf{v}_j||_2^2} \quad \forall\, i \quad , \tag{2}$$

where $\mathbf{v}_i \in \mathbb{R}^3$ are the individual row vectors of the vector feature matrix $\mathbb{R}^{\nu \times 3}$. This vector layer norm has no trainable parameters, but we continue to use normal layer normalization on scalar channels with trainable parameters γ, β.

We study our hypothesis that GVPs augment the geometric reasoning ability of GNNs on a synthetic dataset (Appendix B). The synthetic dataset allows us to control the function underlying the ground-truth label in order to explicitly separate geometric and relational aspects in different tasks. The GVP-GNN matches a CNN on a geometric task and a GNN on a relational task. However, when we combine the two tasks in one objective, the GVP-GNN does significantly better than either a GNN or a CNN.

3.2 Representations of Proteins

The main empirical validation of our architecture is its performance on two real-world tasks: model quality assessment (MQA) and computational protein design (CPD). These tasks, as illustrated in Figure 1B and described in detail in Section 4, are complementary in that one (MQA) predicts a global property and the other (CPD) predicts a property for each amino acid.

We represent a protein structure input as a proximity graph with a minimal number of scalar and vector features to fully specify the 3D structure of the molecule. A protein structure is a sequence of amino acids, where each amino acid consists of four *backbone* atoms[2] and a set of *sidechain* atoms located in 3D Euclidean space. Here we represent only the backbone because our MQA benchmark corresponds to the assessment of backbone structure. In CPD, the sidechains are by definition unknown.

Let X_i be the position of atom X in the ith amino acid. We represent backbone structure as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $\mathfrak{v}_i \in \mathcal{V}$ corresponds to an amino acid and has embedding $\mathbf{h}_\mathfrak{v}^{(i)}$ with the following features:

- Scalar features $\{\sin, \cos\} \circ \{\phi, \psi, \omega\}$, where ϕ, ψ, ω are the dihedral angles computed from C_{i-1}, N_i, $\text{C}\alpha_i$, C_i, and N_{i+1}.
- The *forward* and *reverse* unit vectors in the directions of $\text{C}\alpha_{i+1} - \text{C}\alpha_i$ and $\text{C}\alpha_{i-1} - \text{C}\alpha_i$, respectively.

[1] The nonlinearity σ^+ is a *scaling* by σ^+ applied to the L_2 norm.

[2] $\text{C}\alpha$, C, N, and O.

- The unit vector in the imputed direction of $\mathrm{C}\beta_i - \mathrm{C}\alpha_i$.[3] This is computed by assuming tetrahedral geometry and normalizing

$$\sqrt{\frac{1}{3}}(\mathbf{n} \times \mathbf{c})/||\mathbf{n} \times \mathbf{c}||_2 - \sqrt{\frac{2}{3}}(\mathbf{n} + \mathbf{c})/||\mathbf{n} + \mathbf{c}||_2$$

  where $\mathbf{n} = \mathrm{N}_i - \mathrm{C}\alpha_i$ and $\mathbf{c} = \mathrm{C}_i - \mathrm{C}\alpha_i$. This vector, along with the forward and reverse unit vectors, unambiguously define the orientation of each amino acid residue.
- A one-hot representation of amino acid identity, when available.

The set of edges is $\mathcal{E} = \{\mathbf{e}_{i \to j}\}_{i \neq j}$ for all i, j where $\mathfrak{v}_i$ is among the $k = 30$ nearest neighbors of $\mathfrak{v}_j$ as measured by the distance between their Cα atoms. Each edge has an embedding $\mathbf{h}_e^{(i \to j)}$ with the following features:

- The unit vector in the direction of $\mathrm{C}\alpha_i - \mathrm{C}\alpha_j$.
- The radial basis encoding of the distance $\mathrm{RBF}(||\mathrm{C}\alpha_i - \mathrm{C}\alpha_j||_2)$.
- A sinusoidal encoding of $i - j$ as described in Vaswani et al. (2017), representing distance along the backbone.

In our notation, each feature vector $\mathbf{h}$ is a concatenation of scalar and vector features as described above. Collectively, these features are sufficient for a complete description of the protein backbone.

3.3 Network architecture

Our architecture is a GVP-augmented graph neural network (or GVP-GNN) that takes as input the protein graph defined above and performs graph propagation steps to update the node embeddings:

$$\mathbf{h}_m^{(i \to j)} := g\left(\text{concat}\left(\mathbf{h}_\mathfrak{v}^{(i)}, \mathbf{h}_e^{(i \to j)}\right)\right) \tag{3}$$

$$\mathbf{h}_\mathfrak{v}^{(j)} \leftarrow \text{LayerNorm}\left(\mathbf{h}_\mathfrak{v}^{(j)} + \frac{1}{||\{i : \mathbf{e}_{i \to j} \in E\}||}\text{Dropout}\left(\sum_{i : \mathbf{e}_{i \to j} \in E} \mathbf{h}_m^{(i \to j)}\right)\right) \tag{4}$$

Here, g is a sequence of three GVPs. We do not update edge embeddings and do not use a global graph embedding. Between graph propagation steps, we also use a feed-forward point-wise update layer:

$$\mathbf{h}_\mathfrak{v}^{(i)} \leftarrow \text{LayerNorm}\left(\mathbf{h}_\mathfrak{v}^{(i)} + \text{Dropout}\left(g\left(\mathbf{h}_\mathfrak{v}^{(i)}\right)\right)\right) \tag{5}$$

where g is a sequence of two GVPs. These graph propagation and feed-forward steps update the vector features at each node in addition to its scalar features.

In model quality assessment, we use three graph propagation steps and perform regression against the true quality score of a candidate structure, a global scalar property. To obtain a single global representation, we apply a node-wise GVP to reduce all node embeddings to scalars. We then average the representations across all nodes[4] and apply a final dense feed-forward network to output the network's prediction.

In computational protein design, the network learns a generative model over the space of protein sequences conditioned on the given backbone structure. Following Ingraham et al. (2019), we frame this as an *autoregressive* task and use a masked encoder-decoder architecture to capture the joint distribution over all positions: for each i, the network models the distribution at i based on the complete structure graph, as well as the sequence information at positions $j < i$. The encoder first performs three graph propagation steps on the structural information only. Then, sequence information is added to the graph, and the decoder performs three further graph propagation steps where incoming messages $\mathbf{h}_m^{(i \to j)}$ for $i \geq j$ are computed only with the encoder embeddings. Finally, we use one last GVP with 20-way scalar softmax output to predict the probability of the amino acids.

Further details regarding training and hyperparameters can be found in Appendix D.

[3] Cβ is the second carbon from the carboxyl carbon C.

[4] We also tried learning a weighted average of nodes, but this led to increased overfitting.

4 DATASET AND EVALUATION

In this section, we set up the background and methodology for benchmarking the architecture on model quality assessment and protein design.

Model quality assessment Model quality assessment aims to select the best structural model of a protein from a large pool of candidate structures.[5] The performance of different MQA methods is evaluated every two years in the community-wide Critical Assessment of Structure Prediction (CASP) (Cheng et al., 2019). For a number of recently solved but unreleased structures, called *targets*, structure generation programs produce a large number of candidate structures. MQA methods are evaluated by how well they predict the GDT-TS score of a candidate structure compared to the experimentally solved structure for that target. GDT-TS is a scalar measure of how similar two protein backbones are after global alignment (Zemla et al., 2001).

In addition to accurately predicting the absolute quality of a candidate structure, a good MQA method should also be able to accurately assess the relative model qualities among a pool of candidates for a given target so that the best ones can be selected, perhaps for further refinement. Therefore, MQA methods are commonly evaluated on two metrics: a *global* correlation between the predicted and ground truth scores, pooled across all targets, and the average *per-target* correlation among only the candidate structures for a specific target (Cao & Cheng, 2016; Derevyanko et al., 2018; Pagès et al., 2019). We follow this convention in our experiments.

We train and validate on the set of candidate structures generated in the CASP 5-10 assessments, which collectively contain 528 targets and 79200 candidate structures. For testing, we predict model quality for a total of 20880 stage 1 and stage 2 candidate structures from CASP 11 (84 targets) and 12 (40 targets). Further details on the dataset can be found in Appendix C.

Protein design Computational protein design is the conceptual inverse of structure prediction, aiming to infer an amino acid sequence that will fold into a given structure. In comparison to model quality assessment, computational protein design is more difficult to unambiguously benchmark, as some structures may correspond to a large space of sequences and others may correspond to none at all. Therefore, the proxy metric of *native sequence recovery*—splitting the set of all known native structures in the PDB and attempting to design the sequences corresponding to held-out structures—is typically used to benchmark CPD models (Li et al., 2014; O'Connell et al., 2018; Wang et al., 2018). Drawing an analogy between sequence design and language modelling, Ingraham et al. (2019) also evaluate the model *perplexity* on held-out native sequences. Both metrics rest on the implicit assumption that native sequences are optimized for their structures (Kuhlman & Baker, 2000) and should be assigned high probabilities.

To best approximate real-world applications that may require design of novel structures, the held-out evaluation set should bear minimal similarity to the training structures. We use the CATH 4.2 dataset curated by Ingraham et al. (2019) in which all available structures with 40% nonredudancy are partitioned by their CATH (class, architecture, topology/fold, homologous superfamily) classification. The canonical training, validation, and test splits consist of 18204, 608, and 1120 structures, respectively.

We also report results on TS50, an older test set of 50 native structures first introduced by Li et al. (2014). The smaller size of this benchmark allows a comparison to the computationally expensive physics-based calculations of the `fixbb` protocol in Rosetta, a software suite well-established in the structural biology community (Das & Baker, 2008). No canonical training and validation sets exist for TS50. In order to evaluate on TS50, we remove sequences with more than 30% similarity with any structure in TS50 from our training and validation sets and retrain our model.

5 EXPERIMENTS

Model quality assessment We compare our method against state-of-the-art methods on the CASP 11-12 test set in Table 1. These include the CNN methods 3DCNN (Derevyanko et al., 2018)

[5] We refer to models of protein structure as "structural models" or "candidate structures" to avoid confusion with the term "model" as used in the ML community.

Table 1: Comparison with state-of-the-art methods on CASP 11 and 12 in terms of global (Glob) and mean per-target (Per) Pearson correlation coefficients (higher is better). Each method is classified as one of the three types discussed in Section 2. ProQ3D is set aside as the only method which additionally uses non-structure information. The top performing structure-only method for each metric is in bold, as is the top performing-method overall (if different). Our method generally improves over all other methods.

		CASP 11				CASP 12			
		Stage 1		Stage 2		Stage 1		Stage 2	
Method	Type	Glob	Per	Glob	Per	Glob	Per	Glob	Per
Ours	GNN	**0.84**	**0.66**	**0.87**	**0.45**	**0.79**	**0.73**	**0.82**	**0.62**
3DCNN	CNN	0.59	0.52	0.64	0.40	0.49	0.44	0.61	0.51
Ornate	CNN	0.64	0.47	0.63	0.39	0.55	0.57	0.67	0.49
GraphQA	GNN	0.83	0.63	0.82	0.38	0.72	0.68	0.81	0.61
VoroMQA	Seq	0.69	0.62	0.65	0.42	0.46	0.61	0.61	0.56
SBROD	Seq	0.58	0.65	0.55	0.43	0.37	0.64	0.47	0.61
ProQ3D	Seq	0.80	**0.69**	0.77	0.44	0.67	0.71	0.81	0.60

and Ornate (Pagès et al., 2019), the GNN method GraphQA (Baldassarre et al., 2020), and three methods that use sequential representations—VoroMQA (Olechnovič & Venclovas, 2017), SBROD (Karasikov et al., 2019), and ProQ3D (Uziela et al., 2017). All of these methods learn solely from protein structure,[6] with the exception of ProQ3D, which in addition uses sequence information on related proteins, which is not always available. We include ProQ3D because it is an improved version of the best single-model method in CASP 11 and CASP 12 (Uziela et al., 2017). Our method outperforms all other structural methods in both global and per-target correlation, and even performs better than ProQ3D on all but one benchmark.

We also evaluate DimeNet, a recent 3D-aware GNN architecture which achieves state-of-the-art on many small-molecule tasks (Klicpera et al., 2020), on CASP 11-12. DimeNet differs from our method in that it uses relative edge orientations to indirectly encode geometry into its message-passing operations. While this paradigm appears well-suited for the domain of learning from small molecules, DimeNet does not outperform any of the models in Table 1 on model quality assessment. See Appendix F for detailed results and discussion.

Recent MQA methods have most extensively been benchmarked on the CASP 11-12 datasets. However, for completeness, we also evaluate our method on CASP 13 (Table 2). Because of its recency, many target structures remain publicly unavailable. We use the stage 2 candidate structures of a subset of 20 targets previously used for benchmarking (Baldassarre et al., 2020). Our method achieves improved results over all other methods, including ProteinGCN (Sanyal et al., 2020), a more recent GNN method which was evaluated only on CASP 13. Because of the small sample size, we emphasize that these results, although promising, should be considered preliminary until further structures for CASP 13 are available.

Finally, because our architecture updates vector features along with scalar features at each node embedding, it is possible to visualize *learned* vector features in the intermediate layers of the trained MQA network. We show and discuss the interpretability of such features in Appendix E.

Protein design Our method achieves state-of-the-art performance on CATH 4.2, representing a substantial improvement both in terms of perplexity and sequence recovery over Structured Transformer (Ingraham et al., 2019), a GNN method which was trained using the same training and validation sets (Table 4). Following Ingraham et al. (2019), we report evaluation on short (100 or fewer amino acid residues) and single-chain subsets of the CATH 4.2 test set, containing 94 and 103 proteins, respectively, in addition to the full test set. Although Structured Transformer leverages an attention mechanism on top of a graph-structured representation of proteins, the authors note in ablation studies that removing attention appeared to increase performance. We therefore retrain and

[6]There are two versions of GraphQA; we compare against the one using only structure information.

Table 2: Performance on the 20 publicly available CASP 13 targets, stage 2 in terms of global and mean per-target Pearson correlation. As before, the top structure-only method is in bold, as is the top method overall (if different). Our method outperforms all other methods.

Method	Type	Global	Per-target
Ours	GNN	**0.881**	**0.685**
ProteinGCN	GNN	0.723	0.603
VoroMQA	Seq	0.769	0.665
ProQ3D	Seq	0.849	0.671

Table 3: Relative performance of ablated methods and state-of-the-art GNNs compared with the GVP on sequence recovery (CPD) and mean correlation from CASP 11-12, stage 2 (MQA).

Modification	MQA	CPD
Baseline GVP	100.0%	100.0%
MLP	92.0%	76.1%
Scalars only	94.3%	80.6%
No $\mathbf{W}_h$	96.4%	92.3%
GraphQA	94.0%	–
Structured GNN	–	92.8%

Table 4: Performance on the CATH 4.2 test set and its short and single-chain subsets in terms of per-residue perplexity (lower is better) and recovery (higher is better). Recovery is reported as the median over all structures of the mean recovery of 100 sequences per structure. Our method performs better than Structured Transformer and a variant of it, Structured GNN, in which we replaced the attention mechanisms with standard graph propagation operations (see main text).

		Perplexity			Recovery %		
Method	Type	Short	Single-chain	All	Short	Single-chain	All
Ours	GNN	**7.10**	**7.44**	**5.29**	**32.1**	**32.0**	**40.2**
Structured GNN	GNN	8.31	8.88	6.55	28.4	28.1	37.3
Structured Transformer	GNN	8.54	9.03	6.85	28.3	27.6	36.4

compare against a version of Structured Transformer with the attention layers replaced with standard graph propagation operations (Structured GNN). Our method also improves upon this model.

On the smaller test set TS50, we achieve 44.9% recovery compared to Rosetta's 30% and outperform methods based on each of the three classes of structural representations. Overall, we place 2nd out of 8 methods in terms of recovery (see Appendix F). However, the results for this test set should be taken with a grain of salt, given that the different methods did not use canonical training datasets.

Ablation studies The MQA and CPD methods we have compared against include a number of GNNs (GraphQA, ProteinGCN, Structured Transformer/GNN). We perform ablation studies to identify the aspects of the GVP which most contribute to our improvement over these state-of-the-art models (Table 3). Replacing the GVP with a vanilla MLP layer eliminates the view of geometric information as *geometric vectors*, and propagating only scalar features forces the model to learn scalar-valued, *indirect* representations of geometry. Both modifications result in considerable and similar decreases in performance, underscoring the importance of direct access to geometric information. Eliminating the intermediate transformation $\mathbf{W}_h$ results in a slight but non-negligible decrease in performance. All of these elements contributed to our improvement over Structured GNN (on CPD) and GraphQA (on MQA). See Appendix F for a more detailed discussion.

6 Conclusion

In this work, we developed the first architecture designed specifically for learning on dual relational and geometric representations of 3D macromolecular structure. At its core, our method, GVP-GNN, augments graph neural networks with computationally simple layers that perform expressive geometric reasoning over Euclidean vector features. Our method possesses desirable theoretical properties and empirically outperforms existing architectures on learning quality scores and sequence designs, respectively, from protein structure.

In further work, we hope to apply our architecture to other important structural biology problem areas, including protein complexes, RNA structure, and protein-ligand interactions.

REFERENCES

Namrata Anand, Raphael Ryuichi Eguchi, Alexander Derry, Russ B Altman, and Possu Huang. Protein sequence design with a learned potential. *bioRxiv*, 2020.

Federico Baldassarre, David Menéndez Hurtado, Arne Elofsson, and Hossein Azizpour. GraphQA: Protein model quality assessment using graph convolutional network, 2020. URL `https://openreview.net/forum?id=HyxgBerKwB`.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Jeremy M Berg, John L Tymoczko, and Lubert Stryer. Biochemistry, W.H. Freeman and Company, 2002.

Renzhi Cao and Jianlin Cheng. Protein single-model quality assessment by feature-based probability density functions. *Scientific reports*, 6:23990, 2016.

Sheng Chen, Zhe Sun, Lihua Lin, Zifeng Liu, Xun Liu, Yutian Chong, Yutong Lu, Huiying Zhao, and Yuedong Yang. To improve protein sequence profile prediction through image captioning on pairwise residue distance map. *Journal of Chemical Information and Modeling*, 2019.

Jianlin Cheng, Myong-Ho Choe, Arne Elofsson, Kun-Sop Han, Jie Hou, Ali HA Maghrabi, Liam J McGuffin, David Menéndez-Hurtado, Kliment Olechnovič, Torsten Schwede, et al. Estimation of model accuracy in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1361–1377, 2019.

Nadav Cohen and Amnon Shashua. Inductive bias of deep convolutional networks through pooling geometry. *arXiv preprint arXiv:1605.06743*, 2016.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Rhiju Das and David Baker. Macromolecular modeling with rosetta. *Annu. Rev. Biochem.*, 77: 363–382, 2008.

Georgy Derevyanko, Sergei Grudinin, Yoshua Bengio, and Guillaume Lamoureux. Deep convolutional networks for quality assessment of protein folds. *Bioinformatics*, 34(23):4046–4053, 2018.

Jordan Graves, Jacob Byerly, Eduardo Priego, Naren Makkapati, S Vince Parish, Brenda Medellin, and Monica Berrondo. A review of deep learning methods for antibodies. *Antibodies*, 9(2):12, 2020.

Sharon Hammes-Schiffer and Stephen J Benkovic. Relating protein motion to catalysis. *Annu. Rev. Biochem.*, 75:519–541, 2006.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems*, pp. 15794–15805, 2019.

Mikhail Karasikov, Guillaume Pagès, and Sergei Grudinin. Smooth orientation-dependent scoring function for coarse-grained protein quality assessment. *Bioinformatics*, 35(16):2801–2808, 2019.

Johannes Klicpera, Janek Gro, and Stephan Gnnemann. Directional message passing for molecular graphs, 2020.

Georgii G Krivov, Maxim V Shapovalov, and Roland L Dunbrack Jr. Improved prediction of protein side-chain conformations with scwrl4. *Proteins: Structure, Function, and Bioinformatics*, 77(4): 778–795, 2009.

Brian Kuhlman and David Baker. Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences*, 97(19):10383–10388, 2000.

Zhixiu Li, Yuedong Yang, Eshel Faraggi, Jian Zhan, and Yaoqi Zhou. Direct prediction of profiles of sequences compatible with a protein structure by neural networks with fragment-based local and energy-based nonlocal profiles. *Proteins: Structure, Function, and Bioinformatics*, 82(10): 2565–2573, 2014.

James O'Connell, Zhixiu Li, Jack Hanson, Rhys Heffernan, James Lyons, Kuldip Paliwal, Abdollah Dehzangi, Yuedong Yang, and Yaoqi Zhou. Spin2: Predicting sequence profiles from protein structures using deep neural networks. *Proteins: Structure, Function, and Bioinformatics*, 86(6): 629–633, 2018.

Kliment Olechnovič and Česlovas Venclovas. Voromqa: Assessment of protein structure quality using interatomic contact areas. *Proteins: Structure, Function, and Bioinformatics*, 85(6):1131–1145, 2017.

Guillaume Pagès, Benoit Charmettant, and Sergei Grudinin. Protein model quality assessment using 3d oriented convolutional neural networks. *Bioinformatics*, 35(18):3313–3319, 2019.

Janaina Cruz Pereira, Ernesto Raul Caffarena, and Cicero Nogueira dos Santos. Boosting docking-based virtual screening with deep learning. *Journal of chemical information and modeling*, 56 (12):2495–2506, 2016.

Yifei Qi and John ZH Zhang. Densecpd: Improving the accuracy of neural-network-based computational protein sequence design with densenet. *Journal of Chemical Information and Modeling*, 60(3):1245–1252, 2020.

Soumya Sanyal, Ivan Anishchenko, Anirudh Dagar, David Baker, and Partha Talukdar. Proteingcn: Protein model quality assessment using graph convolutional networks. *BioRxiv*, 2020.

Raphael Townshend, Rishi Bedi, Patricia Suriana, and Ron Dror. End-to-end learning on 3d protein structure for interface prediction. In *Advances in Neural Information Processing Systems*, pp. 15616–15625, 2019.

Karolis Uziela, David Menéndez Hurtado, Nanjiang Shu, Björn Wallner, and Arne Elofsson. Proq3d: improved model quality assessments using deep learning. *Bioinformatics*, 33(10):1578–1580, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Jingxue Wang, Huali Cao, John ZH Zhang, and Yifei Qi. Computational protein design with deep learning neural networks. *Scientific reports*, 8(1):1–9, 2018.

Jonghun Won, Minkyung Baek, Bohdan Monastyrskyy, Andriy Kryshtafovych, and Chaok Seok. Assessment of protein model structure accuracy estimation in casp13: Challenges in the era of deep learning. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1351–1360, 2019.

Adam Zemla, Česlovas Venclovas, John Moult, and Krzysztof Fidelis. Processing and evaluation of predictions in casp4. *Proteins: Structure, Function, and Bioinformatics*, 45(S5):13–21, 2001.

Yuan Zhang, Yang Chen, Chenran Wang, Chun-Chao Lo, Xiuwen Liu, Wei Wu, and Jinfeng Zhang. Prodconn: Protein design using a convolutional neural network. *Proteins: Structure, Function, and Bioinformatics*, 2019.

A Properties of Geometric Vector Perceptrons

A.1 Equivariance and invariance

The vector and scalar outputs of the GVP are equivariant and invariant, respectively, with respect to an arbitrary composition of rotations and reflections in 3D Euclidean space described by R *i.e.*,

$$\text{GVP}((\mathbf{s}, R(\mathbf{V}))) = (\mathbf{s}', R(\mathbf{V}'))$$

Proof. We can write the transformation described by R as multiplying $\mathbf{V}$ with a unitary matrix $\mathbf{U} \in \mathbb{R}^{3\times 3}$ from the right. The L_2-norm, scalar multiplications, and nonlinearities are defined row-wise as in Algorithm 1. We consider scalar and vector outputs separately. The scalar output, as a function of the inputs, is

$$\mathbf{s}' = \sigma\left(\mathbf{W}_m \begin{bmatrix} \|\mathbf{W}_h\mathbf{V}\|_2 \\ \mathbf{s} \end{bmatrix} + \mathbf{b}\right)$$

Since $\|\mathbf{W}_h\mathbf{V}\mathbf{U}\|_2 = \|\mathbf{W}_h\mathbf{V}\|_2$, we conclude $\mathbf{s}'$ is invariant. Similarly the vector output is

$$\mathbf{V}' = \sigma^+\left(\|\mathbf{W}_\mu\mathbf{W}_h\mathbf{V}\|_2\right) \odot \mathbf{W}_\mu\mathbf{W}_h\mathbf{V}$$

The row-wise scaling can also be viewed as left-multiplication by a diagonal matrix $\mathbf{D}$. Since $\|\mathbf{W}_\mu\mathbf{W}_h\mathbf{V}\|_2 = \|\mathbf{W}_\mu\mathbf{W}_h\mathbf{V}\mathbf{U}\|_2$, $\mathbf{D}$ is invariant. Since

$$\mathbf{D}\mathbf{W}_\mu\mathbf{W}_h(\mathbf{V}\mathbf{U}) = (\mathbf{D}\mathbf{W}_\mu\mathbf{W}_h\mathbf{V})\,\mathbf{U}$$

we conclude that $\mathbf{V}'$ is equivariant. □

A.2 THE ABILITY TO APPROXIMATE ROTATION-INVARIANT FUNCTIONS

The GVP inherits an analogue of the Universal Approximation property Cybenko (1989) of standard dense layers. If R describes an arbitrary rotation or reflection in 3D Euclidean space,[7] we show that a GVP can approximate arbitrary scalar-valued functions invariant under R and defined over $\Omega^\nu \subset \mathbb{R}^{\nu\times 3}$, the bounded subset of $\mathbb{R}^{\nu\times 3}$ whose elements can be canonically oriented based on three linearly independent vector entries. Without loss of generality, we assume the first three vector entries can be used.

The machinery corresponding to such approximations corresponds to a GVP G_s with only vector inputs, only scalar outputs, and a sigmoidal nonlinearity σ; followed by a dense layer. This can also be viewed as the sequence of matrix multiplication with $\mathbf{W}_h$, taking the L_2-norm, and a dense network with one hidden layer. Such machinery can be extracted from any two consecutive GVPs (assuming a sigmoidal σ).

Theorem. *Let R describe an arbitrary rotation or reflection in $\mathbb{R}^3$. For $\nu \geq 3$ let $\Omega^\nu \subset \mathbb{R}^{\nu\times 3}$ be the set of all $\mathbf{V} = [\boldsymbol{v}_1, \quad \ldots, \quad \boldsymbol{v}_\nu]^T \in \mathbb{R}^{\nu\times 3}$ such that $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ are linearly independent and $0 < \|\boldsymbol{v}_i\|_2 \leq b$ for all i and some finite $b > 0$. Then for any continuous $F : \Omega^\nu \to \mathbb{R}$ such that $F(R(\mathbf{V})) = F(\mathbf{V})$ and for any $\epsilon > 0$, there exists a form $f(\mathbf{V}) = \mathbf{w}^T G_s(\mathbf{V})$ such that $|F(\mathbf{V}) - f(\mathbf{V})| < \epsilon$ for all $\mathbf{V} \in \Omega$.*

Proof. The idea is to write F as a composition $F = \tilde{F}\circ\omega$ and $\omega = h\circ y$. We show that multiplication with $\mathbf{W}_h$ and and taking the L_2-norm can compute y, and that the dense network with one hidden layer can approximate $\tilde{F}\circ h$.

Call an element $\mathbf{V} \in \Omega^\nu$ *oriented* if $\boldsymbol{v}_1 = x_1\mathbf{e}_x$, $\boldsymbol{v}_2 = x_2\mathbf{e}_x + y_2\mathbf{e}_y$, and $\boldsymbol{v}_3 = x_3\mathbf{e}_x + y_3\mathbf{e}_y + z_3\mathbf{e}_z$, with $x_1, y_2, z_3 > 0$. Define $\omega : \Omega^\nu \to \mathbb{R}^{3\nu-3}$ to be the *orientation function* that orients its input and then extracts the vector of $3\nu - 3$ coefficients, $[x_1, x_2, y_2, x_3, y_3, z_3, \ldots, x_i, y_i, z_i, \ldots]^T$. These elements can be written as

$$\begin{aligned}
x_1 &= \|\boldsymbol{v}_1\|_2 \\
x_i &= \boldsymbol{v}_i \cdot \boldsymbol{v}_1 / x_1, \quad i \geq 2 \\
y_2 &= \sqrt{\|\boldsymbol{v}_2\|_2^2 - x_2^2} \\
y_i &= (\boldsymbol{v}_i \cdot \boldsymbol{v}_2 - x_i x_2)/y_2, \quad i \geq 3 \\
z_3 &= \sqrt{\|\mathbf{v}_3\|_2^2 - x_3^2 - y_3^2} \\
z_i &= (\boldsymbol{v}_i \cdot \boldsymbol{v}_3 - x_i x_3 - y_i y_3)/z_3, \quad i \geq 4
\end{aligned}$$

and are invariant under rotation and reflection, because they are defined using only the norms and inner products of the $\mathbf{v}_i$. Then $F = \tilde{F}\circ\omega$, where $\tilde{F} : [-b, b]^{3\nu-3} \to \mathbb{R}$.

[7] More precisely, if R describes a unitary transformation.

The key insight is that if we construct $\mathbf{W}_h$ such that the rows of $\mathbf{W}_h\mathbf{V}$ are the original vectors $\mathbf{v}_i, \forall i$ and all differences $\mathbf{v}_i - \mathbf{v}_j, \forall i, j \leq \min(i, 3)$, then we can compute $\omega(\mathbf{V})$ from the row-wise norms of $\mathbf{W}_h\mathbf{V}$. That is, $\omega = h \circ y$ where $\mathbf{y} = y(\mathbf{V}) = \|\cdot\|_2 \odot (\mathbf{W}_h\mathbf{V}) \in \mathbb{R}^{4\nu-6}$ and h is an application of the cosine law. The GVP precisely computes $\mathbf{y}$ as an intermediate step: $G_s(\mathbf{V}) = \sigma \odot (\mathbf{W}_m\mathbf{y} + \mathbf{b})$. It remains to show that there exists a form $\tilde{f}(\mathbf{y}) = \mathbf{w}^T[\sigma \odot (\mathbf{W}_h\mathbf{y} + \mathbf{b})]$ that ϵ-approximates $\tilde{F} \circ h : [-2b, 2b]^{4\nu-6} \to \mathbb{R}$. Up to a translation and uniform scaling of the hypercube, this is the result of the Universal Approximation Theorem (Cybenko, 1989). □

B SYNTHETIC TASKS

We perform controlled experiments on a synthetic dataset in order to analyze the benefits of the GVP architecture and determine if it indeed improves the geometric and joint geometric-relational reasoning abilities of GNNs.

Dataset The synthetic dataset is designed to mimic the essential qualities of the domain of protein structures. Each "structure" consists of $n = 100$ random points in $\mathbb{R}^3$, distributed uniformly in the ball of radius $r = 10$, with the constraint that no two points are less than distance $d = 2$ apart. Each position is also associated with a random unit vector (a "sidechain") to endow it with an orientation. Three points are randomly chosen and are labelled as "special"; these will be used to define the learning tasks. We generate 20k "structures" and split them 80% train : 10% validation : 10% test.

In the voxelized representation of a data point, the volume is voxelized into unit cubes. Each point is partitioned into a neighborhood of eight voxels by trilinear interpolation, such that exact coordinate information is retained. Separate channels are used for the special and non-special points. The "sidechains" are represented with a set of $n = 100$ points located at the ends of the unit vectors and mapped into a third channel.

In the graph-structured representation of a data point, a proximity graph is drawn with $k = 10$ nearest neighbors. Each node is labelled with a one-hot encoding of its type ("special" or "non-special") and each edge with its Euclidean length. In the vanilla GNN, orientation information is encoded by additionally including all three dot products in each edge embedding. In the GVP-GNN, each node embedding contains the node's "sidechain" vector, and each edge embedding contains a unit vector indicating the direction of the edge.

Tasks We identify two regression tasks to exemplify geometric and relational reasoning, respectively. In the "Off-center" task, the network predicts the distance from the centroid of the three special points to the centroid of the entire structure. In the "Perimeter" task, the network predicts the perimeter of the triangle defined by the three special points. We characterize the former as primarily geometric, as it requires reasoning about global properties of the 3D *shape*, and the latter as primarily relational, as it involves distances between three *pairs* of nodes. Finally, to represent a problem with geometric *and* relational aspects, in the "Combined" task we attempt to predict the difference of the (normalized) off-center and perimeter objectives.

Models We train a 3-layer shallow CNN and a 3-layer GNN with a single-layer feed-forward network. These are compared against a GVP-GNN that is otherwise identical to the standard GNN. To reflect the spirit of the synthetic experiment, all models have the same intermediate dimensionality of 32 (4 vector and 20 scalar channels in the GVP-GNN), we use the same training procedure for all models, and no hyperparameter tuning or architecture search is performed.

Results The results of the synthetic experiments are shown in Table 5. The vanilla GNN significantly outperforms the CNN on the perimeter task, while the CNN significantly outperforms the GNN on the off-center task, supporting our conceptual framework of the relative strengths of the two architectures. However, the GVP-GNN matches (and even outperforms) the CNN on the geometric task while maintaining the GNN's performance on the relational task. It additionally significantly outperforms *both* models on the combined task. On the basis of these results, the GVP appears successful in combining the strengths of the CNN and GNN into a single architecture.

Table 5: Performance of the three compared model architectures on the off-center (geometric), perimeter (relational), and combined objectives. The MSE losses are standardized such that predicting a constant value (i.e. the mean) would result in unit loss. Results are reported as the mean $\pm$ S.D. over $k = 5$ randomly shuffled splits, where the best of three random seeds is taken for each split.

Model	Parameters	Off-center (geometric)	Perimeter (relational)	Combined
CNN	59k	0.319 ± 0.014	0.532 ± 0.028	0.522 ± 0.016
GNN	40k	0.871 ± 0.045	0.128 ± 0.009	0.421 ± 0.025
GVP-GNN	22k	$\mathbf{0.206 \pm 0.024}$	$\mathbf{0.106 \pm 0.006}$	$\mathbf{0.155 \pm 0.024}$

C DATASETS

Model quality assessment For training and validation, we use 528 targets from CASP 5-10 and 150 candidate structures per target. These targets are partitioned at random into 480 training targets and 48 validation targets. We include native structures for training and validation to make use of the greatest range of GDT-TS scores available.

In the CASP assessments, stage 1 refers to a set of 20 candidate structures per target and stage 2 to a set of 150 candidate structures per target (5 from each structure prediction server). Both sets are pre-designated by the CASP organizers.

There has been slight inconsistency in the literature with regards to the exact composition of the CASP 11 and 12 test sets. We use the list established by Karasikov et al. (2019) because nearly all recent methods have been benchmarked on this set at some point. We do not include native structures for testing to best mimic CASP and real-world application scenarios and because other methods were not tested on native structures.

All structures are relaxed with SCWRL4 (Krivov et al., 2009) to eliminate artifacts from the various prediction servers. The exact numbers of targets and structures in each set can be found in Table 6.

Table 6: MQA datasets

Dataset	# Targets	# Structures	Includes natives?
Training	480	72000	Yes
Validation	48	7200	Yes
CASP 11 stage 1	84	1680	No
CASP 11 stage 2	83	12450	No
CASP 12 stage 1	40	800	No
CASP 12 stage 2	40	5950	No
CASP 13 stage 2	20	2999	No

Protein design As described in the main text, we use the CATH 4.2 dataset and splits as curated by Ingraham et al. (2019) and the TS50 test set as curated by Li et al. (2014). To evaluate on TS50, we remove sequences with more than 30% similarity as computed by `PSIBLAST` to any protein in TS50 from CATH 4.2.

D TRAINING AND HYPERPARAMETERS

To train the MQA model to perform regression against the model quality score, we use a sum of an absolute loss and a pairwise loss. That is, for each training step we intake pairs i, j where i, j are candidate structures for the same target and compute

$$\mathscr{L} = H(y^{(i)} - \hat{y}^{(i)}) + H(y^{(j)} - \hat{y}^{(j)}) + H\left((y^{(i)} - y^{(j)}) - (\hat{y}^{(i)} - \hat{y}^{(j)})\right) \tag{6}$$

where H is the Huber loss. When reshuffling at the beginning of each epoch, we also randomly pair up the candidate structures for each target. Interestingly, adding the pairwise term also improves

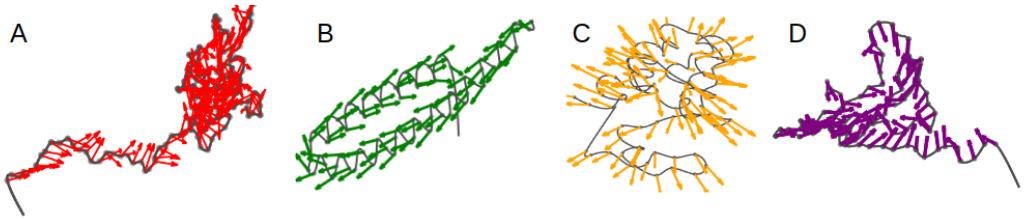

Figure 2: Four different learned vector channels of the MQA model are visualized on four separate structures. The backbone is represented as a chain of points, and each vector is rooted at the position of the amino acid node to which it belongs. From left to right: the vectors appear to A) point in the direction of motion that would make the protein more compact; B) point along the central axis of the alpha helix; C) point outwards from the structure; and D) point inwards into the structure.

global correlation, likely because the much larger number of possible pairs makes it more difficult to overfit.

To train the CPD model to perform classification / discrete generative modelling, we use the cross-entropy / negative log likelihood loss. We also experimented with label smoothing, but this did not improve performance.

For both the MQA and CPD model, we use node and hidden embeddings with 16 vector and 100 scalar channels and edge embeddings with 1 vector and 32 scalar channels. The input node and edge features are first transformed by a sequence of GVPs to these dimensionalities before graph propagation. In all training runs, we use the Adam optimizer to perform mini-batch gradient descent. Batches are constructed by grouping structures of similar size to have a maximum of 1800 residues per batch for CPD and 3000 residues per batch for MQA. We also tune the following hyperparameters over a total of 70 training runs:

- Learning rate in the range of 10^{-4} to 10^{-3}
- Dropout probability in the range of 10^{-4} to 10^{-1}
- Number of graph propagation layers in the range of 3 to 6
- Relative weight of the MQA pairwise loss in the range of 0 to 2

All models are implemented in TensorFlow 2.1 and trained for a maximum of 100 epochs. This takes around two days for both models on a single Titan X GPU. However, we note that the GPU memory, not compute power, is the bottleneck when training, based on the the average volatile GPU usage. We therefore anticipate that the runtime can be further optimized.

E Visualization and Interpretation of Learned Features

The geometric vector perceptron updates the vector features, in addition to scalar features, at node embeddings during graph propagation. Therefore, while the *input* vector channels represent the forward and reverse directions at each amino acid, the *intermediate* layers represent learned vector features. Could some of these features correspond to interpretable properties of the structure? Among a total of 64 intermediate vector channels learned by the MQA model, a few appeared visually interpretable and are shown on selected structures in Figure 2. We caution against generalizing from the necessarily small number of images that could be manually inspected, but find these preliminary visualizations intriguing.

F Additional Results

F.1 DimeNet on MQA

DimeNet (Klicpera et al., 2020) is a recent GNN architecture designed to incorporate the 3D geometry of small molecule graphs by updating edge embeddings with incoming messages from neighboring edges, where the relative orientation of each edge is encoded in a local spherical Bessel

basis. DimeNet and our architecture are similar in that both seek to leverage geometric aspects of a problem domain on top of graph-structured representations. However, unlike our architecture, it specifically updates *edge* embeddings and must represent each pair of neighboring edges, and therefore does not scale well to large protein structure graphs. In evaluating DimeNet on MQA, we could only extend the distance cutoff to 7.5 angstroms, while 30 neighbors corresponds to roughly 13 angstroms. DimeNet does not perform comparably to our model, or to previous GNNs designed for learning from structure such as GraphQA. (Table 7).

Table 7: Comparison of our GVP architecture, DimeNet, and the GraphQA, another GNN-based MQA method, on CASP 11-12. As in the main text, the global and mean per-target Pearson correlations are shown. DimeNet does not perform comparably to either our method or GraphQA.

	CASP 11 Stage 2		CASP 12 Stage 2	
Method	Global	Per-target	Global	Per-target
Ours	**0.87**	**0.45**	**0.82**	**0.62**
GraphQA	0.82	0.38	0.81	0.61
DimeNet	0.61	0.30	0.62	0.47

F.2 CPD: Results on TS50 test set

A number of methods for computational protein design only report results for the test set TS50 and not the CATH 4.2 test set. Our comparison with these methods is shown in Table 8. No canonical training and validation sets exist for TS50. In order to evaluate on TS50, we remove sequences with more than 30% similarity from the CATH 4.2 training and validation sets and retrain our model. The methods to which we compare include two CNNs (ProDCoNN and DenseCPD), a distance-map method (SBROF), and sequential representation methods (Wang's model and SPIN2). We outperform all other methods with the exception of DenseCPD, a CNN method with canonical orientations. Interestingly, DenseCPD leverages the same underlying representation as ProDCoNN, yet achieves remarkably better performance. The main difference between the two methods is that ProDCoNN has 4 convolutional layers and DenseCPD 21 layers organized into dense residual blocks.

Table 8: Sequence recovery on TS50. Recovery for our method is a median over all proteins of the mean recovery of 100 designs per protein; recovery for other methods, which model residues independently, is just classification accuracy. Our method is the second best-performing, behind the CNN method DenseCPD. There is no canonical training and validation set for methods evaluated on TS50.

Method	Recovery %
Ours	44.9
ProDCoNN (Zhang et al., 2019)	40.7
DenseCPD (Qi & Zhang, 2020)	**50.7**
SBROF (Chen et al., 2019)	39.2
Wang's model (Wang et al., 2018)	33.0
SPIN2 (O'Connell et al., 2018)	33.6
SPIN (Li et al., 2014)	30.3
Rosetta	30.0

F.3 Ablation studies

In order to determine which aspects of the GVP contribute most to our improvement over existing state-of-the-art GNNs, we train a number of ablated models for MQA and CPD and compare them against the baseline GVP models. In addition to Table 3 in the main text, we report a more detailed breakdown of performance metrics in Table 9. Replacing the GVP with a vanilla MLP layer or propagating only scalar features both remove direct access to geometric information, hurting the performance of the model. Propagating only the vector features results in an even more significant

decrease as it both eliminates important scalar input features (such as torsion angles and amino acid identity) and the part of the GVP with approximation guarantees. Therefore, the dual scalar/vector design of the GVP is essential. Finally, eliminating the intermediate transformation $\mathbf{W}_h$, and instead taking the norms of the final transformed vectors $\mathbf{V}_\mu$, results in a slight but non-negligible decrease in performance. The results suggest that each aspect of the GVP contributed to our improvement over state-of-the-art GNNs.

Table 9: Performance of ablated models and other state-of-the-art GNNs (Structured GNN for CPD; GraphQA for MQA) on both tasks. Metrics for MQA are reported as global and mean per-target Pearson correlations. Perplexity and recovery are defined in the same way as in Table 4.

	MQA				CPD	
	CASP 11 Stage 2		CASP 12 Stage 2		CATH4.2 All	
Modification	Global	Per-target	Global	Per-target	Perplexity	Recovery
Baseline GVP	**0.87**	**0.45**	0.82	**0.62**	**5.29**	**40.2**
MLP layer	0.84	0.36	0.79	0.59	7.76	30.6
Only scalars	0.84	0.38	**0.83**	0.59	7.31	32.4
Only vectors	0.56	0.16	0.57	0.39	11.05	23.2
No $\mathbf{W}_h$	0.86	0.41	0.81	0.60	5.85	37.1
GraphQA	0.82	0.38	0.81	0.61	–	–
Structured GNN	–	–	–	–	6.55	37.3