Compressing the Latent Space of Single-Sequence Protein Predictors for Multimodal Generation

Anonymous Authors

Abstract

ESMFold learns a joint latent space of sequence and structure while requiring only sequence as input. However, the latent space of ESMFold is disorganized and we find pathologies, similar to those observed in large language models, that render these models unusable for multimodal representation learning. Meanwhile, latent diffusion in both continuous and discrete spaces have improved efficiency and performance in image and multimodal generation, but are built on an abundance of knowledge on autoencoders for images. To create a protein encoder which captures structural and functional information for generative modeling in the latent space, we create CHEAP (Compressed Hourglass Embedding Adaptations of Proteins) representations, and find that the channel dimension of ESMFold latent spaces can be compressed by up to $256 \times$ while retaining rich structural, sequence, and functional information, as demonstrated on protein understanding benchmarks and reconstruction performance.

1. Introduction

Generative modeling has emerged as a popular tool for protein design due to its scaling properties on complex data distributions (Watson et al., 2023; Ingraham et al., 2022). To synthesize the molecule in the lab, however, a sequence that can fold into the structure must be specified. Despite the dual importance of structure and sequence modalities, existing methods are typically single modality, and generate either structure (Watson et al., 2023; Ingraham et al., 2022) or sequence (Gruver et al., 2023; Alamdari et al., 2023). By sampling sequence and structure simultaneously, one gains structure-conditioned control over protein design, which is highly useful given the large array of structure-mediated use cases in drug discovery, such as efficient hit binding, targeting specific biological pathways, learning protein-protein interactions (PPI), and perform more efficient docking.

Sequence-to-structure prediction (Jumper et al., 2021; Lin et al., 2023) have been posited as "protein structure foundation models" (Wang et al., 2024). ESMFold (Lin et al., 2023) demonstrates that sequence-to-structure prediction can be built on top of protein language model (pLM) embeddings. Intriguingly, at inference time, though pLM attentions capture pairwise contact information (Rao et al., 2021), the pairwise input to the structural module is initialized to zero (Section C and Figure 2). All the information required for the structure, therefore, is contained within this sequence embedding. Thus, by learning a generative model to approximate the distribution of natural proteins under this representation, one can perform simultaneous multimodal generation of structure and sequence. Importantly, this allows structural diffusion from only sequence as inputs, which is desirable because sequence datasets can be $10^2$ to $10^4$ times larger than structural datasets.

Naively intercepting this latent space, however, presents numerous challenges. The latent space of large language models (LLMs) often have high activations in certain chan-
We also demonstrate how compression affects reconstruction performance and function prediction across both discrete and continuous compression schemes.

2. Related Works

Latent Space Generation in Visual Media Latent-space based generative models is often used to manage the high-dimensional nature of visual data; design of these successful methods is built on ample research around architectural and algorithmic choices for visual representations. Contemporary scalable generative models for vision and multimodal media often fall into two categories: those working with discrete representations in either an masked-token or autoregressive next-token prediction manner (He et al., 2022; Bao et al., 2021; Chang et al., 2022; Yu et al., 2022; Razavi et al., 2019; Villegas et al., 2022; Esser et al., 2021), or diffusion-based models with continuous data (Ho et al., 2020; Saharia et al., 2022; Rombach et al., 2022; Peebles & Xie, 2023; Ho et al., 2022).

Protein diffusion and multimodal generation Though protein structure diffusion has seen empirical and lab-verified success (Watson et al., 2023; Bennett et al., 2024), such models learn a probability distribution over plausible protein structures, rather than the joint distribution of both sequence and structure. Such models rely on an exogenous structure-to-sequence prediction step to obtain the sequence. Empirical results show that such methods often exhibit “low designability”, where generated structures may not have a sequence that can fold into that structure. Some works attempt to generate both structure and sequence simultaneously, usually alternating between sequence-to-structure and structure-to-sequence steps (Lisanza et al., 2023; Chu et al., 2023).

Massive Activations in LLMs Large transformers often suffer from the massive activations (Sun et al., 2024) or outlier features (Dettmers et al., 2022) phenomenon, where output values in intermediate layers exhibit unusually high values on the magnitude of up to 20x larger. Sun et al. (2024) provides detail study and finds that for both Llama and ViT, finding channels which have outlier values, and dominate attention patterns. In contrast to the well-tamed latent space of two-stage latent diffusion works in images (Rombach et al., 2022), the latent space of LLMs should be expected to be much more unwieldy, and we indeedly find this to be true empirically (Figures 2).
Compressing the Latent Space of Single-Sequence Protein Predictors for Multimodal Generation

Figure 4. Comparing FSQ and VQ-VAE performance across a range of codebook sizes. Consistent with reported findings in Mentzer et al. (2023), performance is initially higher for VQ-VAE, with FSQ outperforming VQ-VAE at codebook sizes greater than $2^{10}$. Blue arrows denote metrics where lower is better, and red arrows denote metrics where higher is better.

Figure 5. Examining codebook utilization for FSQ and VQVAE. VQVAE suffers from codebook collapse, while FSQ is constructed by design to use all codes. Codebook utilization is generally more favorable with FSQ (top).

3. Methods

3.1. Organizing the Latent Space for Generation

Per-Channel Normalization  To address the issue of massive activations as shown in Figure 2, we use a per-channel normalization scheme. The embeddings are processed as:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \times \left( c_{\text{max}} - c_{\text{min}} \right) + c_{\text{min}}$$

where $x_{\text{min}}$ and $x_{\text{max}}$ are vectors with shape (1024,) broadcasted along the length dimension to match $x$, and denote statistics calculated for each channel, independently. This prevents outlier channel values from dominating the normalization. For consistency with image diffusion works, we choose $c_{\text{min}} = -1$ and $c_{\text{max}} = 1$.

Latent Space Compression with the Hourglass Compression Transformer  Unlike images, proteins have different lengths, which precludes usage of convolution-based autoencoders. However, we reason that the downsampling operation in convolution neural network may also be key for compressing information from adjacent amino acids into local motifs. Furthermore, on transformer-based generative models, reducing the length dimension also helps with managing the quadratic memory requirements of transformer attention layers (Vaswani et al., 2017). Therefore, we choose an encoder architecture inspired by the Hourglass Transformer (Nawrot et al., 2021), which includes a shortening operation, $g(x)$, that transforms a tensor $x$ with shape $(L, D)$ to $(\frac{L}{2}, D)$. The Hourglass Compression Transformer architecture is described in Algorithm A.

3.2. Compression Representations

For discrete representation, we further examine two schemes: (1) Vector-quantized variational auto-encoders (VQ-VAE) (Van Den Oord et al., 2017) and (2) Finite Scalar Quantization (FSQ) (Mentzer et al., 2023). The VQ-VAE (Van Den Oord et al., 2017) learns a discrete representation of the input, typically of images. In the forward pass, the encoder $h_e$ produces a continuous feature representation of input $x$. Then, each feature vector is mapped to a discrete code in the codebook space, $C$, where each discrete code is associated with a continous vector $e_i$. The complete VQ-VAE loss is:

$$L_{\text{VQ}} = \log p(x|h_q(z)) + ||\text{sg}(h_e(x)) - z||^2 + \beta ||h_e(x) - \text{sg}(z)||^2$$

VQ-VAE can be prone to “codebook collapse”, whereby a few codes are over-utilized, especially for larger codebook sizes (Takida et al., 2022; Łańcucki et al., 2020; Dhariwal et al., 2020; Huh et al., 2023). We therefore also investigate using the FSQ (Mentzer et al., 2023) approach. Rather than using a nearest-neighbor search to choose a code, FSQ directly quantizes the continuous encoder representations $z \in \mathbb{R}^d$ into $L$ bins:
Compressing the Latent Space of Single-Sequence Protein Predictors for Multimodal Generation

Figure 6. Qualitative examination of structure reconstruction $\phi_\Omega 1(x_{\text{reconstructed}})$, for different bottleneck $c$ dimensions (original embedding contains 1024 channels), and a shortening factor of 2. Despite aggressive downsampling and channel down projection, substantial structure information can be recovered via neural compression.

$$z = h_e(x), z \in \mathbb{R}^d$$  
Encoder output

$$\hat{z} = \tanh(z)$$  
Bound to $[-1, 1]$

$$\tilde{z} = \text{round}(\lfloor (L/2) \rfloor \cdot \hat{z})$$  
Discretize to $L$ bins

The predetermined bins $L$ is selected to be small relative to VQ-VAE codebook sizes. The implicit codebook size $|C|$, however, comes from the combinatorial possibilities arising from using one of $L$ integers at each of the $D$ channels. For $z$ with $d$ channels, there are $d$ associated integer representations, and thus $|C| = L^d$. A large implicit codebook can thus be achieved, while forcing all codes to be used.

4. Results

Similar to using perceptual loss evaluation in addition to reconstruction performance, we also examine reconstruction performance in sequence and structure space (Figure 2). Template-modeling score (TM-Score) is a backbone only metric of structure reconstruction, while root-mean-square deviation (RMSD) is a more fine-grained measure between atom positions. Sequence reconstruction accuracy examines token matches after decoding back to sequence space.

4.1. Discrete Compression

Though reference experiments exist in images with regard to how big codebook sizes should be, it is unclear how many bits of information can be expected from a joint representation of both sequence and all-atom structure. We therefore do a thorough investigation across different codebook sizes for both VQVAE and FSQ. Consistent with findings in Mentzer et al. (2023), we find that FSQ outperforms VQVAE for codebook sizes larger than $2^{10}$ across both reconstruction MSE and performance measured in structure and sequence spaces.

4.2. Continuous Compression

Table 4.2 examines performance on benchmarks from Xu et al. (2022). The downprojected version of the latent that is intercepted upstream of the original ESM output performs competitively or better than ESM1b, despite aggressive compression. More benchmark results can be found in the Appendix. Figure 4.2 demonstrates that good backbone alignment (i.e. TM-Score), RMSD below idealized inter-residue bond lengths, and near-perfect sequence reconstruction performance can be retained even after aggressive compression.

5. Conclusion

CHEAP embeddings investigate the compression of the ESMFold latent space for protein multimodal generation, to both enable speed and flexibility. Using an Hourglass autoencoder architecture, our results demonstrate that functional and structural information learned by ESMFold can be compactly captured.

<table>
<thead>
<tr>
<th># Dimensions</th>
<th>Cont</th>
<th>Fold</th>
<th>SSP</th>
<th>Yst</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDE</td>
<td>400</td>
<td>0.10</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>Moran</td>
<td>240</td>
<td>0.07</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>LSTM</td>
<td>640</td>
<td>0.26</td>
<td>0.69</td>
<td>0.54</td>
</tr>
<tr>
<td>Transformer</td>
<td>512</td>
<td>0.18</td>
<td>0.60</td>
<td>0.54</td>
</tr>
<tr>
<td>CNN</td>
<td>21</td>
<td>0.10</td>
<td>0.66</td>
<td>0.55</td>
</tr>
<tr>
<td>ResNet</td>
<td>512</td>
<td>0.20</td>
<td>0.70</td>
<td>0.49</td>
</tr>
<tr>
<td>ProtBert</td>
<td>1024</td>
<td>0.40</td>
<td>0.82</td>
<td>0.54</td>
</tr>
<tr>
<td>ESM-1b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHEAP (ours)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.28</td>
<td>0.15</td>
<td>0.82</td>
<td>0.45</td>
</tr>
<tr>
<td>64</td>
<td>0.42</td>
<td>0.45</td>
<td>0.85</td>
<td>0.48</td>
</tr>
<tr>
<td>128</td>
<td>0.38</td>
<td>0.47</td>
<td>0.85</td>
<td>0.51</td>
</tr>
<tr>
<td>256</td>
<td>0.23</td>
<td>0.50</td>
<td>0.85</td>
<td>0.51</td>
</tr>
<tr>
<td>512</td>
<td>0.37</td>
<td>0.53</td>
<td>0.86</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 1. Comparing representation learning results on benchmarks described in Xu et al. (2022). CHEAP performs competitively or better despite aggressive compression.
Figure 7. (Top) Comparing TM-Score and sequence recovery of compressed structure and the original prediction for compressed representations. (Bottom) Comparing the RMSD with RMSPD (super-imposition free) to distinguish reconstruction errors in orientation only vs. those which also alter pairwise distances.
A. Hourglass Compression Transformer

Though different operations may be used for $g(x)$, we find a simple linear downsampling to work well (Algorithm ??), which can be seen as a convolution with a filter size and stride both equal to $S$. Since the original model is designed for sequence-to-sequence tasks rather than compression, we remove the skip connections that would make the solving the reconstruction task trivial. Additionally, we add a projection layer along the channel dimension after each shortening operation. At training time, $x_{\text{reconstructed}}$ is used for calculating the mean-squared-error reconstruction loss, and at inference time, the output of the encoder is used as the compressed representation, with additional processing in the bottleneck, depending on if the compression is discrete or continuous (Section 3.2).

Algorithm 1 Hourglass Compression Transformer

```
embedding $\mathbf{a} \leftarrow \mathbf{a} \in \mathbb{R}^{L \times D}$,
mask $\mathbf{m} \leftarrow \mathbf{m}^{\text{Length}([L])} := \{1, 0\}^{L}$,
shortening factor $S \leftarrow S \in \mathbb{Z}$,
downprojection factor $K \leftarrow K \in \mathbb{Z}$,
downprojection $W_d \leftarrow W_d \in \mathbb{R}^{D \times \frac{D}{K}}$,
upprojection $W_u \leftarrow W_u \in \mathbb{R}^{D \times \frac{D}{K}}$,
$a, m \leftarrow \text{Pad length to multiple of } S$
$a \leftarrow \text{Transformer}(a, m)$
$a' \leftarrow \text{LinearDownsampling}(a, S)$
$m' \leftarrow \sum_S m^{\text{Length}([L]) \rightarrow \frac{S}{K}} > 0 \{\text{Reduce}\}$
$a' \leftarrow \text{AttentionResampling}(a', a, m')$
c $\leftarrow W_d a'$
if quantize then
c $\leftarrow \text{Tanh}(c)$
else
c $\leftarrow \text{Bottleneck}(c, m')$
end if
$a' \leftarrow W_u c$
a $\leftarrow \text{LinearUpsampling}(c', m')$
m $\leftarrow m^{\text{Length}([K]) \rightarrow \frac{S}{K}} > 0 \{\text{Repeat}\}$
a $\leftarrow \text{AttentionResampling}(a, a', m)$
a $\text{reconstructed} \leftarrow \text{Transformer}(a, m)$
return: $a_{\text{reconstructed}} \in \mathbb{R}^{L \times D}, c \in \mathbb{R}^{\frac{D}{K} \times \frac{D}{K}}$
```

B. Discrete Representation Learning

The assembled array of learned codes and their vector embeddings $\mathbf{z} = \{\mathbf{e}_1, \mathbf{e}_2, ... \mathbf{e}_{|C|}\}$ and their corresponding feature vectors are fed into the decoder $h_q(\mathbf{z})$. Since the quantization operation is not differentiable, the straight-through estimator (STE) () is used by copying the gradients from the decoder input to the encoder output. The codebook is selected via a nearest-neighbor search in Euclidean space; auxiliary losses are introduced to pull the codeword vectors towards the unquantized encoder outputs. As in autoencoder training, a reconstruction loss between output and input is also used.

C. Defining a Joint Structure-Sequence Latent Space

A key observation for this work is that during inference use, it is empirically sufficient to initialize the pairwise representation input $\mathbf{z}$ as an array of zeros, and thus all information needed for structure is contained in $\mathbf{x} = \phi_s(\mathbf{s})$ (Figure 2). The core idea of the PLAID framework is to train a generative model $p_\theta(\mathbf{x})$ to characterize the joint latent space of all feasible protein sequence and structures as $\phi(s, \Omega)$, as defined by the intermediate layers of ESMFold (Lin et al., 2023). We intercept the sequence representation that is the direct input into the folding trunk (Figure 2).

Constructing forward- and backward-mappings To define mappings from sequence $s$ and structure $\Omega$ to the joint multimodal representation space $\phi(s, \Omega)$, we can decompose $\phi(s, \Omega) = \phi_s(\cdot) \circ \phi_\Omega(\cdot)$, and use components of ESMFold to
Compressing the Latent Space of Single-Sequence Protein Predictors for Multimodal Generation

represent \( \phi_\Omega(\cdot) \) and \( \phi_s(\cdot) \) mappings:

\[
\begin{align*}
x &= \phi_s(s) & \text{ESM2 Language Model} \\
\Omega &= \phi_\Omega^{-1}(x, \varnothing) & \text{ESMFold Structure Module}
\end{align*}
\] (1) (2)

At inference time, after sampling \( \tilde{x} \sim p_\theta(x) = p_\theta(s, \Omega) \), we can generate new protein sequences as \( \tilde{s} = \phi_\Omega^{-1}(\tilde{x}) \), which we see from Eq. 1 is the backward mapping of the ESM2 language model. This “back-mapping” sequence decoder can be trained separately, and since the space is already the output of a language model, we observe a per-token accuracy of 99.7% on a randomly partitioned heldout set.

D. Further Benchmark Results

<table>
<thead>
<tr>
<th># Dimensions</th>
<th>Flu ↑</th>
<th>Sta ↑</th>
<th>( \beta )-lac ↑</th>
<th>Sol ↑</th>
<th>Sub ↑</th>
<th>Bin ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDE</td>
<td>400</td>
<td>0.64</td>
<td>0.65</td>
<td>0.60</td>
<td>0.49</td>
<td>0.77</td>
</tr>
<tr>
<td>Moran</td>
<td>240</td>
<td>0.40</td>
<td>0.32</td>
<td>0.38</td>
<td>0.58</td>
<td>0.31</td>
</tr>
<tr>
<td>LSTM</td>
<td>640</td>
<td>0.49</td>
<td>0.53</td>
<td>0.14</td>
<td>0.70</td>
<td>0.63</td>
</tr>
<tr>
<td>Transformer</td>
<td>512</td>
<td>0.64</td>
<td>0.65</td>
<td>0.26</td>
<td>0.70</td>
<td>0.56</td>
</tr>
<tr>
<td>CNN</td>
<td>21</td>
<td>0.68</td>
<td>0.64</td>
<td>0.78</td>
<td>0.64</td>
<td>0.59</td>
</tr>
<tr>
<td>ResNet</td>
<td>512</td>
<td>0.64</td>
<td>0.13</td>
<td>0.15</td>
<td>0.67</td>
<td>0.52</td>
</tr>
<tr>
<td>ProtBert</td>
<td>1024</td>
<td>0.34</td>
<td>0.70</td>
<td>0.62</td>
<td>0.59</td>
<td>0.59</td>
</tr>
<tr>
<td>ESM-1b</td>
<td>1280</td>
<td>0.43</td>
<td><strong>0.75</strong></td>
<td>0.53</td>
<td>0.67</td>
<td><strong>0.80</strong></td>
</tr>
<tr>
<td><strong>CHEAP (ours)</strong></td>
<td>4</td>
<td>0.14</td>
<td>0.40</td>
<td>0.13</td>
<td>0.60</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.22</td>
<td>0.44</td>
<td>0.17</td>
<td>0.64</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.27</td>
<td>0.55</td>
<td>0.23</td>
<td>0.65</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.28</td>
<td>0.56</td>
<td>0.28</td>
<td>0.67</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.31</td>
<td>0.56</td>
<td>0.28</td>
<td>0.69</td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>0.41</td>
<td>0.58</td>
<td>0.38</td>
<td>0.70</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>0.47</td>
<td>0.60</td>
<td>0.41</td>
<td>0.71</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>0.51</td>
<td>0.63</td>
<td>0.36</td>
<td><strong>0.72</strong></td>
<td>0.74</td>
</tr>
<tr>
<td><strong>No compression</strong></td>
<td>1024</td>
<td>0.52</td>
<td>0.64</td>
<td>0.45</td>
<td><strong>0.72</strong></td>
<td>0.76</td>
</tr>
</tbody>
</table>

Table 2. Benchmarks on function and localization.

References


Compressing the Latent Space of Single-Sequence Protein Predictors for Multimodal Generation


Compressing the Latent Space of Single-Sequence Protein Predictors for Multimodal Generation


