NLP final project

Maria Bochenek
University of Warsaw
m.bochenek@student.uw.edu.pl

Natalia Rutecka
University of Warsaw
n.rutecka@student.uw.edu.pl

Małgorzata Sudoł
University of Warsaw
mj.sudol@student.uw.edu.pl

Supervised by:
Grzegorz Preibisch
University of Warsaw

Abstract

The advancement of robust natural language models has increased the ability to learn meaningful representations of protein sequences. Deep transformer-based autoencoders such as ReLSO can be trained to jointly generate sequences as well as predict their fitness due to their highly structured latent space. However training and deploying this model can be costly due to its use of the standard self-attention mechanism. To address this, we propose LinReLSO, a model that incorporates the ReLSO architecture with Linformer self-attention, and we evaluate its performance in comparison to the original architecture. Our findings demonstrate that LinReLSO not only consumes less resources and speeds up computations but also surpasses the original model in terms of both reconstruction and prediction accuracy. LinReLSO is publicly available at https://github.com/talkana/ReLSO-with-Linformer.

1 Introduction

In recent years, natural language processing (NLP) has emerged as a powerful tool in the field of bioinformatics, revolutionizing the way we extract knowledge and gain insights from vast amounts of textual biological data, such as DNA or protein sequences. Its applications span from predicting the localization of binding sites and various motifs in the genome (Ji et al. (2020), Qiao et al. (2022)) to predicting cancer from gene expression profiles Khan and Lee (2023), as well as protein function annotation, structure prediction, protein generation, and drug design (Cao and Shen (2021), Rives et al. (2021), Ferruz et al. (2022), Castro et al. (2022), Grechishnikova (2021)). Protein generation based on deep neural models is a promising approach that has the potential to expand the space of protein design beyond natural proteins. It enables us to discover novel amino acid sequences that can fold into desired spatial structures Verkuil et al. (2022) or exhibit desired properties such as stability, catalysis, binding affinity, or fluorescence Castro et al. (2022). Having an efficient model that facilitates the search for such sequences will have a significant impact on therapeutic use and other fields focused on maximizing specific protein features. Using neural networks for protein generation may overcome the primary challenge of sequence-based protein design, which is the vast and complex space of possibilities that extends far beyond the reach of modern throughput technologies.

Recent approaches to sequence-based protein generation employ Transformer-based Vaswani et al. (2017) encoder-decoder models with additional latent space optimization to address the problem of finding sequences with high fitness (protein sequences with desired properties) Castro et al. (2022). One of the challenges faced by Transformer-based models, is the computational efficiency of self-attention, whose time- and space-complexity scales quadratically with the number of tokens (here: amino acid letters) in the input sequence.

By reducing self-attention complexity, we can obtain additional computational resources, which may enable us to train models at lower cost or apply more sophisticated models using the same amount of resources.

In this paper we introduce an optimized variant of the Transformer-based encoder proposed in Castro et al. (2022). The optimized variant features a reduced embedding size and utilizes a linear attention mechanism based on the approach proposed in Wang et al. (2020), addressing the computational challenges associated with self-attention. To evaluate the performance of our optimized model, we conduct experiments on a benchmark TAPE dataset Sarkisyan et al. (2016), which contains domain sequences from green fluorescent protein (GFP) together with their fitness measured in log fluorescence level. We evaluate the predictions of the

optimized model and the achieved reductions in time and space complexity. Moreover, we conduct latent space optimisations to generate unseen protein sequence with maximised log fluorescence levels.

2 Related work

In recent years, numerous approaches have been proposed to incorporate deep learning into the process of finding protein sequences with optimal fitness. However, many methods focus on predicting the fitness of a given protein sequence alone (Brookes and Listgarten (2020), Brookes et al. (2019), Yang et al. (2019), Linder and Seelig (2021)) and lack the ability to directly generate sequences with maximized fitness, limiting their potential for automating the screening process.

One of the deep learning architectures most widely used in proteomics is the transformer model Shuang Zhang et al. (2023). A pioneering work by Vig et al. (2021) has shown high performance and interpretability of transformer-based models on protein sequences. Notably, attention has been shown to capture the structural and biophysical properties of proteins, including the position of the binding sites - the key functional components of proteins. These properties have been leveraged to pretrain a number of protein language models (Brandes et al. (2022), Rives et al. (2021), Rao et al. (2019), Elnaggar et al. (2021)).

To address the generation of protein sequences with enhanced fitness, Castro et al. (2022) introduced a novel transformer-based autoencoder model called ReLSO. ReLSO incorporates latent space optimization and is trained not only to reconstruct protein sequences but also to predict their fitness from the latent representation. The regularization techniques employed by ReLSO facilitate smooth interpolation in the latent space and robust generation of (novel) protein sequences.

Similarly to many protein transformer models, ReLSO uses Scaled Dot-Product Attention as in Vaswani et al. (2017) that makes room for improvement when it comes to computing attention more efficiently. In recent years several sub-quadratic methods have been proposed to speed up self-attention mechanisms. One of the first ones was Sparse Attention used in Sparse Transformer (Child et al., 2019). It reduces complexity to $O(n\sqrt{n})$ by only considering a subset of the computations in the self-attention matrix.

Another approach to addressing the complexity of self-attention is the BigBird attention mechanism, presented by Zaheer et al. (2021). Specifically designed to handle long input sequences, BigBird incorporates sparse attention, consisting of three components: *global attention*, where a set of g global tokens attends to all parts of the sequence; *sliding window attention*, where all tokens attend to a set of w local neighboring tokens; and *random attention*, where all tokens attend to a set of r random tokens. BigBird has demonstrated high performance on various benchmark tasks, including both natural language processing (NLP) and computational biology problems.

Finally, another method to reduce the complexity of self-attention is Linformer, a matrix factorization (decomposition) technique introduced by Wang et al. (2020). Linformer approximates the attention dot-product matrix by its k-rank factorization, reducing the attention complexity to $O(k * n)$. Compared to the standard self-attention mechanism, Linformer introduces only one additional hyperparameter, k. This characteristic is particularly important in our study, as a large number of parameters can hinder the attainment of conclusive results when comparing different model architectures. Moreover, since ReLSO needs to be trained separately for each protein dataset to predict distinct fitness values, it is advantageous to employ a simpler architecture with fewer hyperparameters.

3 Model

3.1 Architecture

The original ReLSO model consists of four fundamental components: a transformer-based encoder, an additional bottleneck network, a convolutional decoder, and a feedforward fitness prediction network.

Encoder The encoder of the ReLSO model comprises an embedding layer and a sequence of stacked transformer-like multi-head attention encoder blocks. The input sequence, represented as one-hot encodings of amino acids, undergoes token-level embedding and is then processed through the multi-head attention blocks.

Given that enhancing the performance of the encoder network was the primary objective of our research, we will delve deeper into its architecture in subsequent sections of this paper.

Bottleneck network Subsequent to the encoder, the encoded output is forwarded through a singu-

lar layer referred to as the "bottleneck network". This network performs a projection of the encoder output onto the final latent space representation (denoted as z) for an input sequence x.

The combination of the *encoder* and *bottleneck network* is denoted by Castro et al. (2022) as f_θ.

Decoder The decoder network (referenced as g_θ) is constructed with a linear layer followed by a series of convolutions with batch normalizations and ReLU activations in between. The role of the decoder network is to translate latent space points into amino acid sequences, (by design, the model is restricted to generate sequences of fixed length). As we do not introduce any changes in the decoder in our work, we kindly direct the reader to the original work for a more comprehensive understanding of the decoder's details.

Fitness prediction head The fitness prediction head, denoted as h_θ, is comprised of two linear layers with batch normalization between them and a ReLU activation at the top of the network. The primary objective of the h_θ network is to estimate the fitness level associated with a given sequence based on its latent representation.

3.2 Reconstruction loss and fitness loss

The combined networks f_θ, g_θ, and h_θ, trained without additional regularizations, are collectively referred to as the Jointly-Trained Auto-Encoder (JT-AE) by Castro et al. (2022). The loss function employed for training the JT-AE consists of a summation of the reconstruction loss, which stems from the decoding task, and the regression loss, associated with the fitness prediction task. In the original paper, these losses are defined as follows:

$$\begin{aligned} L_{rec} &= \|g_\theta(f_\theta(\mathbf{x})) - -\mathbf{x}\|, \\ L_{reg} &= \|h_\theta(f_\theta(\mathbf{x})) - -\mathbf{y}\|, \\ L_{JTAE} &= L_{rec} + L_{reg}. \end{aligned} \tag{1}$$

Where L_{rec} stands for reconstruction loss, L_{reg} is MSE regression loss, $\mathbf{x}$ is the original sequence and $\mathbf{y}$ is fitness value of $\mathbf{x}$.

Comment - ambiguities with $\mathbf{L_{JT-AE}}$ loss

Although the paper presents formulas with both losses weighted equally (with a weight of one), it is more natural to assign appropriate weights to both loss components when formulating the multitask loss. Interestingly, despite the inclusion of the multitask loss formula twice in the paper, the actual code weights both losses as described in (2).

In our model, we also apply appropriate weighting to all contributing losses within the general loss function.

$$\begin{aligned} L_{JT-AE} = \gamma\, L_{rec} + \alpha\, L_{reg}, \\ \gamma, \alpha \in \mathcal{R} \end{aligned} \tag{2}$$

The second concern pertains to the representation of the reconstruction loss. The formula in the paper utilizes a "norm on sequences of letters," but there is no explicit clarification of how this norm should be interpreted (perhaps akin to the Hamming distance?).

Furthermore, in the code implementation, the reconstruction loss is computed using token-level cross-entropy, where the one-hot encoding vector is compared with the resulting distribution over the vocabulary for each position in the sequence. In our model, we also adopt the cross-entropy loss as the only defined version and a commonly used loss function for such tasks.

In summary, the JT-AE loss utilized in our model, and for training ReLSO, is as follows:

$$\begin{aligned} L_{rec} &= CE(\mathbf{x}_{one\_hot}, \hat{\mathbf{x}}_{softmax}), \\ L_{reg} &= \|h_\theta(f_\theta(\mathbf{x})) - -\mathbf{y}\|, \\ L_{JT-AE} &= \gamma\, L_{rec} + \alpha\, L_{reg}, \end{aligned} \tag{3}$$

where $CE(\,\cdot\,,\,\cdot\,)$ is a position-wise sum of cross entropies for sequence of one-hot encodings $\mathbf{x}_{one\_hot}$ and distributions $\hat{\mathbf{x}}_{softmax}$ returned by a network and the rest is as described above.

3.3 Regularization losses

In order to achieve a more structured latent space fitness landscape, facilitating the effective generation of novel sequences with desired fitness levels from the latent space, two additional loss components are added to the the JT-AE loss in the training procedure.

3.3.1 Regularization for latent space pseudoconcavity

This regularization addresses a fundamental challenge in latent space optimization. It is often observed that the optimization trajectory may deviate significantly from the training data, leading to a degradation in model prediction accuracy and yielding unreliable outcomes (Castro et al. (2022)).

To mitigate this issue and promote more reliable gradient-based optimization on the latent fitness landscape, a bias towards regions in the latent space that are close to the training data is introduced

during the fitness function learning process. For each latent representation z within the batch, a set of n points $z_i, i = 1, .., m$ is sampled from regions situated far from z. These z_i points are assigned low fitness values and incorporated into the fitness prediction loss.

For a single sequence $\mathbf{x}$ the negative loss term is:

$$L_{neg\_z} = \frac{1}{M} \sum_{i=1}^{m} (y_{neg\_i} - min(Y)), \quad (4)$$

where m denotes number of negative samples z_i per sequence, y_{neg_i} is the value predicted by h_θ for i-th negative sample and $min(Y)$ is the minimum observed fitness in the dataset.

3.3.2 Regularization for latent space continuity

The purpose of this regularization is to achieve a desirable level of "smoothness" within the latent space. Given a pair of sequences x_1 and x_2, along with their respective latent representations z_1 and z_2, we aim for the latent point z_3, lying on the segment spanned by z_1 and z_2, to be encoded to a sequence $\hat{x}_3$ that resides "between" the two sequences in a sense of sequence editing distance (Hamming distance).

The loss for interpolating between a single pair of points is:

$$L_{itrp} = \sum_{i=1}^{k} max(0, r_k), \quad (5)$$

where:

$$r_k = \frac{\|\hat{\mathbf{x}}_\mathbf{1} - \hat{\mathbf{x}}_\mathbf{3}\| + \|\hat{\mathbf{x}}_\mathbf{2} - \hat{\mathbf{x}}_\mathbf{3}\|}{2} - \|\hat{\mathbf{x}}_\mathbf{1} - \hat{\mathbf{x}}_\mathbf{2}.\|$$

For each batch the subset of points z_1 is sampled, and for each z_1 and its k nearest neigbours z_2, the parwise interpolations z_3 are computed and the losses from these interpolations reconstructions are incorporated in the total loss sum.

Comment - ambiguities with L_{neg} loss

There is an ambiguity in the definition of this loss in the ReLSO paper, as the specific norm used is not explicitly defined in the paper. However, unlike the reconstruction loss, the notation in this case does not represent cross-entropy loss. Upon analyzing the code, it is apparent that the p-norm with p=1 is applied to the original token one-hot encodings and the network outputs, previously discretized to one-hot representations. Essentially, this computation can be understood as a roundabout way of calculating the Hamming distance multiplied by 2.

In our work, where the focus was primarily on modifying only the attention computation components and thoroughly reproducing the network, we retained the original code implementation for this loss. However, for future work, it is advisable to consider revising this fragment for improved code readability and better description.

4 Contribution: improvements in encoder complexity

The original ReLSO model utilizes a transformer-based encoder with multi-head dot-product attention. However, the transformer attention mechanism suffers from a significant drawback, which is its quadratic time and space complexity. This becomes a bottleneck when training transformer models, particularly for datasets with long sequences.

Training time is particularly crucial in the context of ReLSO, as a new model needs to be trained from scratch for each protein and its fitness optimization. Additionally, a new training run is required whenever generative abilities of models based on data from different experiments are to be compared. Furthermore, the average length of proteins in Eukaryota is reported to be 472 amino acids (Tiessen et al., 2012), indicating that the sequence length used to train the network will be, in most cases, on average, 2-20 times larger than the data used and described by Castro et al. (GFP domain is 20 aa long and sequences in TAPE are 237 aa long). To address these issues, we enhance the model's time and space complexity by introducing a modified encoder architecture that employs the Linformer (Wang et al., 2020) attention mechanism. In the subsequent sections of this paper, we will refer to our model as LinReLSO.

4.1 LinReLSO encoder

In each encoder block, we replace the quadratic multi-head attention with Linformer attention (Wang et al., 2020). To accomplish this, we add two additional projection matrices, E_i and F_i, to each encoder block, which project the sequence of embeddings of length n onto a sequence s' of length $k < n$ (where k is a hyperparameter of the model). Subsequently, we project these sequences onto k key vectors and k $value$ vectors accordingly, and apply self dot-product attention between

sets of n *query* vectors and k key' vectors. Using this attention matrix , we compute a convex combination of k $value'$ vectors, resulting in the final v" vector for each head.

This modification enables the modified attention operation to be performed with a time and space complexity of $O(n)$ with respect to the sequence length (see Figure 1).

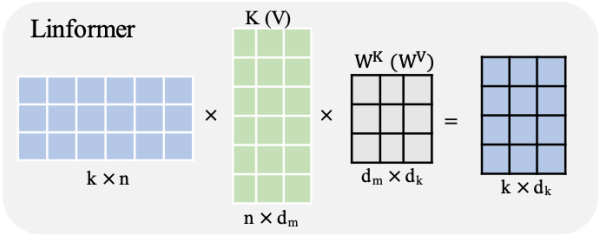


Figure 1: Linformer attention from (Wang et al., 2020)

Our approach allows for more efficient allocation of computational resources. We use separate E_i and F_i matrices for keys and values, different in each head and each encoder block.

5 Experimental set-up

The objective of our experimentation was to modify the attention computation while keeping the remaining parts of the model as close to the original design as possible. This approach allows for reliable comparisons to be made regarding the impact of switching from the original attention mechanism to the Linformer attention on the model's performance.

As we encountered several differences between the descriptions provided in the paper and the implementation in the attached code, there were instances where ambiguity arose regarding the exact architecture. In each such case, we document the discrepancy in our description below and clarify the choice we made. In most instances, we opted to maintain consistency with the version implemented in the code, as it appeared to better suit the problem at hand.

5.1 Models

We trained six models with different parameters to facilitate a comparison between ReLSO (models 1 and 4) and LinReLSO (molels 2, 3, 5, 6).

model	1	2	3	4	5	6
embedding_dim	20			100		
k	-	25	60	-	25	60

For LinReLSO two values of Linformer's k hyperparameter were chosen: $k = 60$ ($\approx 0.25\cdot$ of sequence length (237)) inspired by k/n ratio used in Wang et al. (2020) and $k = 25$ as to check whether further reducing dimensionality would result in drop in performance.

The *embedding_dim* parameter denotes the dimensionality of the token embeddings multiplied by the number of attention heads in each encoder block (number of heads equals 4 in the ReLSO model).

Loss function

We trained our model using a multitask loss:

$$\mathcal{L} = \gamma\mathcal{L}_{rec} + \alpha\mathcal{L}_{reg} + \eta\mathcal{L}_{neg\_z} + \lambda\mathcal{L}_{intrp},$$

where $\mathcal{L}_{rec}$, $\mathcal{L}_{reg}$. $\mathcal{L}_{neg\_z}$, $\mathcal{L}_{intrp}$ are losses defined in section 3, calculated for a batch.
Weights we used were: $\gamma = 1$, $\alpha = 1$, $\eta = 0.8$, $\lambda = 0.01$.

Reducing token embedding dimension

In the original architecture, each amino acid is represented by a one-hot encoding vector of length 22 (21 positions for amino acids and one for the padding token). Encoding such vectors in a space with dim = 25 (*embedding_dim* = 100, heads=4) is computationally inefficient, as it wastes computational resources.
To address this and check how it influences the model training and performance, we trained additional ReLSO and LinReLSO models with a reduced *embedding_dim* of 20, which maintains the necessary encoding capacity but also does not introduce more significant changes to the architecture.

Inconsistencies in the original architecture descriptions

There were some inconsistencies between the original paper and code regarding the number of encoder blocks, default embedding dimension, and hidden dimension used in the model.

According to the paper, the model should have ten encoder blocks, an *embedding_dim* of 300, and a *hidden_dim* of 400.

However, the code implementation uses *six* encoder blocks, a default *embedding_dim* of 100 (which represents the actual embedding dimension multiplied by the number of heads in a single encoder block, that means that for a single head, its input-output dimension equals 25), and a *hidden_dim* of 200.

In our study, we decided to follow the version presented in the code, with $N = 6$ encoder blocks, an $embedding\_dim$ of 100, and a $hidden\_dim$ of 200. This decision was made to maintain consistency with the original transformer architecture (Vaswani et al. (2017)) and to utilize a lower dimensionality, which we believe is more suitable for this problem. We also acknowledge that even smaller values for $embedding\_dim$ and $hidden\_dim$ may be more appropriate for this task.

5.2 Dataset

We conducted our experiments on the TAPE dataset (Sarkisyan et al., 2016), as this is one of two datasets with the longest sequences among four datasets used by ReLSO authors, and it is also a well known benchmark dataset for protein modeling. TAPE dataset contains 54025 protein sequences of length 237, derived from green fluorescent protein (GFP) and the fitness measured is log level of fluorescence. We used original train-test-validation split $(21446 - 27217 - 5362)$ as ReLSO authors did, to enable comparisons of the results.

5.3 Training details

Training of each model was performed for 300,000 steps (400 epochs) with a learning rate of 0.00002 on one GPU (4GB NVIDIA GeForce RTX 3050 Laptop GPU) using a batch size of 64. To incorporate negative sampling and interpolative sampling regularizations, 32 artificial latent points are sampled during each forward pass, resulting in an effective batch size of 128. Probability of dropout in encoder module was set to 0.2, while the latent dimension size was set to 30 for all models.

5.4 Protein sequence optimization

To evaluate the protein generation capability of our trained models, we employ gradient ascent as a measurement technique. Initially, we select 10 sequences randomly from the training dataset. Subsequently, using a given trained model, we encode each input sequence to derive its latent representation, resulting in 10 points within the latent space for each trained model, referred to as initial points.

For each initial point, denoted as x, we calculate the gradient of its fitness value predicted by the trained model. This gradient is then utilized to update the latent point. In our experiments, we repeat this process for a total of $n = 200$ steps with a learning rate of 0.013, following the hyperparameters used by Castro et al. (2022). In the end, a final latent point z is obtained. Subsequently, this point is decoded to generate the optimized sequence s_z and the optimized fitness f_z is predicted.

Overall, we generate 10 pairs (s_z, f_z) for each of the 6 trained models, resulting in a total number of 60 pairs.

6 Results

Several metrics were used to evaluate and compare models' performance. The AE loss (cross-entropy loss measured during training between input sequences and output reconstructions) was tracked during training to evaluate the reconstruction ability of the Autoencoder (Figure 8). Initially, ReLSO with $embedding\_dim = 20$ achieved lower loss values than other models, but by the end of training, all LinReLSO models exhibited lower AE loss values than the classic ReLSO. LinReLSO models with $k = 60$ showed a larger drop in AE loss values during training compared to those with $k = 25$, but the differences became negligible towards the end of training. To further evaluate the AE's reconstruction capability for protein sequences, Cross-entropy Loss was computed between input sequences and reconstructed sequences (3). Consequently we found that LinReLSO models consistently obtained lower CE values across all datasets in comparison with classic ReLSO. The differences between LinReLSO models were minimal, particularly on the test dataset, indicating good generalization of the models. The ReLSO model with $embedding\_dim = 20$ displayed the poorest reconstruction performance out of all models.

To evaluate fitness predictions we computed the following metrics between target fitness values and predicted fitness values: Pearson correlation coefficient (Figure 4), MSE Loss (Figure 5) and L1 Loss (Figure 6). We observed stronger linear correlation between fitness targets and predictions across all datasets in LinReLSO models in comparison to the ReLSO models.

LinReLSO with $k = 60$ outperformed LinReLSO with $k = 25$ on the training and validation datasets but slightly underperformed on the test dataset. Among all LinReLSO models, LinReLSO with $embedding\_dim = 20$ and $k = 25$ exhibited the closest similarity to classic ReLSO models in terms of MSE and L1 loss values on training and validation dataset. However, both metrics demonstrated higher values for the original ReLSO model, suggesting a weaker ability to predict fitness.

Memory usage and wall time were additional metrics used to compare the models. Although comparing wall time was somewhat challenging, we noticed that training LinReLSO models was approximately 2 hours faster comparing to original 8 and 6 hours for ReLSO with $embedding\_dim = 100$ and $embedding\_dim = 20$ respectively.

The examination of process memory usage (Figure 10) revealed that all LinReLSO models utilized less memory than ReLSO models. It can be observed that reducing the $embedding\_dim$ had a more pronounced effect on reducing memory usage compared to reducing the value of k. Note that we only reference our limited experimental setup.

We also utilized predicted fitness values of the reconstructed optimal sequences to compare the protein generation capabilities of the six trained models. The distributions of these fitness values are illustrated in Figure 2.

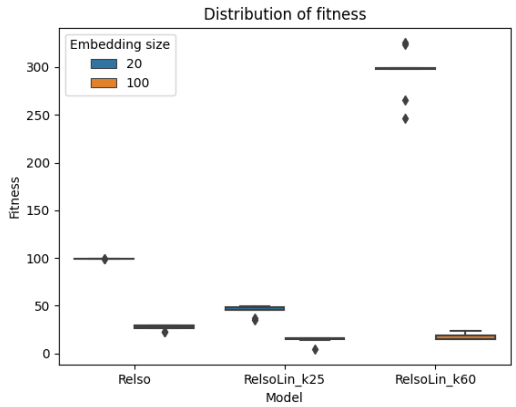


Figure 2: Distribution of fitness values inferred from latent space optimizations using 10 starting points.

Our observations reveal significant variations in the obtained fitness values across the models, while demonstrating relatively similar values among the 10 initial points within each model. Notably, an enhancement in fitness values is evident when employing an embedding size of 20 across all three tested architectures. Among the models, the LinReLSO model with k=60 achieves the highest fitness. Overall, our evaluation indicates that high fitness sequences are more easily discovered within less complex models. These findings suggest that simpler architectures may be better suited for the task of optimizing latent fitness landscape for the high fitness levels.

7 Discussion

By incorporating Linformer attention as a replacement for the conventional self-attention mechanism, our LinReLSO model achieved significant improvements in reducing memory usage and accelerating computations. Across all experimental setups, LinReLSO consistently outperformed the standard ReLSO model, not only in terms of conserving resources but also in its predictive capabilities. The saved resources can be used to handle datasets containing longer protein sequences or to support the expansion of the model.

The results highlight LinReLSO's superior performance in both sequence reconstruction and fitness prediction across all the datasets used. It is noteworthy that LinReLSO outperforms ReLSO in all measured metrics, particularly on the test dataset, indicating its superior generalization ability. The strong linear correlation observed between fitness targets and predictions in LinReLSO suggests that although the learned patterns may be simpler, they result in improved generalization. The enhanced accuracy on unseen or out-of-distribution examples, such as the test dataset, can be attributed to the change in attention mechanism, which encourages the model to learn more robust and generalized representations of the input data.

To further evaluate the impact of Linformer's attention on the model's performance, expanding the experimental setup is necessary. This expansion would involve testing the models on all four benchmark datasets used in Castro et al. (2022). Additionally, future work should include experiments with different embedding sizes, as they have a significant influence on performance metrics. However, it should be noted that adjusting embedding sizes would require further modifications to the model architecture, as the currently allowed sizes are constrained by the number of attention heads (e.g., $embedding\_dim / num\_heads = 0$).

To gain a more comprehensive understanding of our model's protein generation ability, it would be beneficial to carefully analyze per-position distributions of the returned optimized sequences. While the model was trained to minimize the loss between the input and output autoencoder sequences, it remains unclear whether the output sequences can be unambiguously transformed into valid protein sequences, specifically whether the vectors in the decoded sequences closely resemble the one-hot encoding of a single amino acid. Furthermore, re-

peating the sampling of latent space points with different seed values could provide insights into the robustness of the presented results.

Overall, our findings suggest that architectures using lower-dimensional representations may be more suitable for optimizing latent space for finding representations with optimal fitness, considering both computational efficiency and generative ability. However, further experiments are needed to assess whether these findings can be generalized to different protein datasets.

8 Conclusions

Transformer models are notoriously slow to train and deploy in practice, thus every way of saving computational resources is valuable. In this paper we focus on the ways of improving the computational complexity of the transformer-based model proposed by Castro et al. (2022). We examined the impact of reducing embedding dimension from 100 to 20 and using Linformer attention instead of standard quadratic dot-product self-attention. We trained and compared performance of six models: two architectures used linformer attention with k=25, two used linformer attention with k=60 and two - regular dot-product attention. Each version was trained with 2 embedding sizes: 20 and 100. Overall, we compared the computational performance, predictions and generation ability of the trained models on a benchmark TAPE dataset.

Our findings indicate that changing attention mechanisms from standard Transformer self-attention to Linformer self-attention speeds up computations and decreases memory usage as well as increases reconstruction and prediction accuracy. Additionally, the modification of the attention mechanism enhances the model's ability to generalize to unseen and out-of-distribution data. Furthermore, we observed that a smaller embedding size improves optimal sequence generation ability of our model. However, further investigation is required to validate the influence of embedding size on predictions.

Our results provide a strong foundation for further investigations aimed at reducing complexity and redundancy in generative models for protein sequences.

9 References

References

Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. 2022. ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. 2019. Conditioning by adaptive sampling for robust design. In *Proceedings of the 36th International Conference on Machine Learning*, pages 773–782. PMLR. ISSN: 2640-3498.

David H. Brookes and Jennifer Listgarten. 2020. Design by adaptive sampling. ArXiv:1810.03714 [cs, q-bio, stat].

Yue Cao and Yang Shen. 2021. TALE: Transformer-based protein function Annotation with joint sequence–Label Embedding. *Bioinformatics*, 37(18):2825–2833.

Egbert Castro, Abhinav Godavarthi, Julian Rubinfien, Kevin B. Givechian, Dhananjay Bhaskar, and Smita Krishnaswamy. 2022. ReLSO: A Transformer-based Model for Latent Space Optimization and Generation of Proteins. ArXiv:2201.09948 [cs].

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. ArXiv:1904.10509 [cs, stat].

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. 2021. ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning. Pages: 2020.07.12.199554 Section: New Results.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. 2022. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348.

Daria Grechishnikova. 2021. Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Scientific Reports*, 11(1):321.

Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V. Davuluri. 2020. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Pages: 2020.09.17.301879 Section: New Results.

Anwar Khan and Boreom Lee. 2023. DeepGene Transformer: Transformer for the gene expression-based classification of cancer subtypes. *Expert Systems with Applications*, 226:120047.

Johannes Linder and Georg Seelig. 2021. Fast differentiable DNA and protein sequence optimization for molecular design. *BMC Bioinformatics*, 22(1):510. ArXiv:2005.11275 [cs, stat].

Yanhua Qiao, Xiaolei Zhu, and Haipeng Gong. 2022. BERT-Kcr: prediction of lysine crotonylation sites by a transfer learning method with pre-trained BERT models. *Bioinformatics (Oxford, England)*, 38(3):648–654.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S. Song. 2019. Evaluating Protein Transfer Learning with TAPE. *Advances in Neural Information Processing Systems*, 32:9689–9701.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. 2021. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 118(15):e2016239118.

Karen S. Sarkisyan, Dmitry A. Bolotin, Margarita V. Meer, Dinara R. Usmanova, Alexander S. Mishin, George V. Sharonov, Dmitry N. Ivankov, Nina G. Bozhanova, Mikhail S. Baranov, Onuralp Soylemez, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor A. Kondrashov. 2016. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401.

Shuang Zhang, Rui Fan, Yuti Liu, Shuang Chen, Qiao Liu, and Wanwen Zeng. 2023. Applications of transformer-based language models in bioinformatics: a survey. *Bioinformatics Advances*. S2ID: 7389b6ebbf36f4d869a02e305e2ef52ad2c92264.

Axel Tiessen, Paulino Pérez-Rodríguez, and Luis José Delaye-Arredondo. 2012. Mathematical modeling and comparison of protein size distribution in different plant, animal, fungal and microbial species reveals a negative correlation between protein size and protein number, thus providing insight into the evolution of proteomes. *BMC Research Notes*, 5(1):85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Robert Verkuil, Ori Kabeli, Yilun Du, Basile Wicky, Lukas Milles, Justas Dauparas, David Baker, Sergey Ovchinnikov, Tom Sercu, and Alexander Rives. 2022. Language models generalize beyond natural protein. Technical report. Type: article.

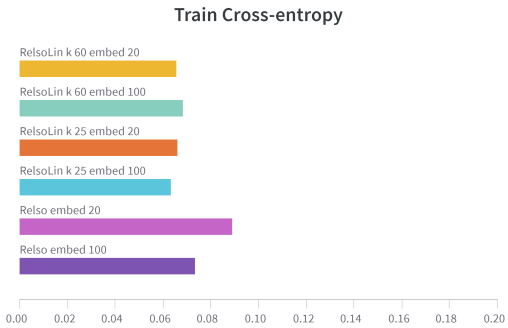
Jesse Vig, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. BERTology Meets Biology: Interpreting Attention in Protein Language Models. ArXiv:2006.15222 [cs, q-bio].

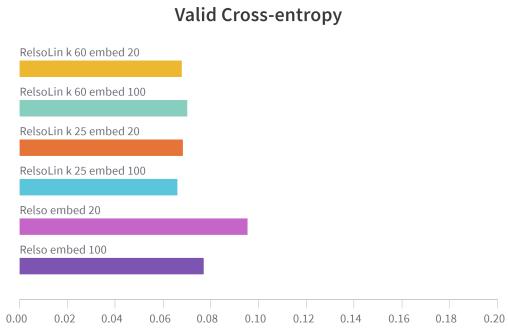
Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. ArXiv:2006.04768 [cs, stat].

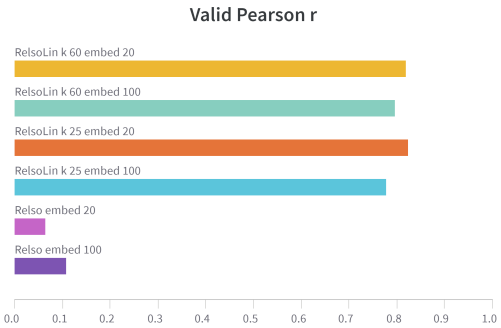
Kevin K Yang, Zachary Wu, and Frances H Arnold. 2019. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694.

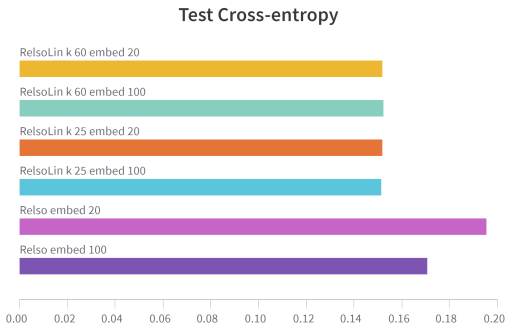
Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big Bird: Transformers for Longer Sequences. ArXiv:2007.14062 [cs, stat].

A Additional plots









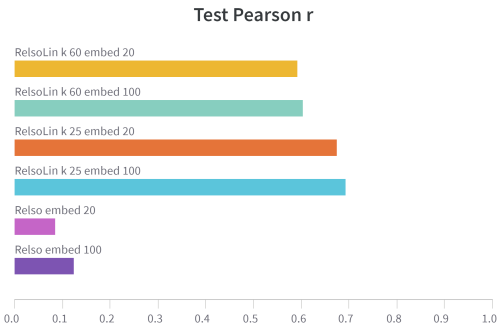
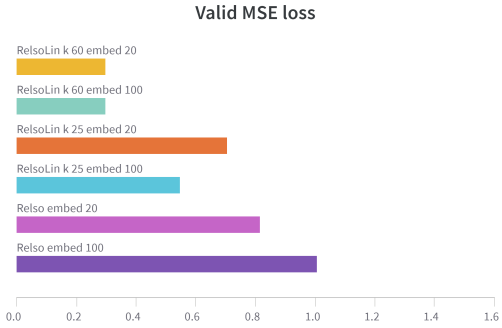
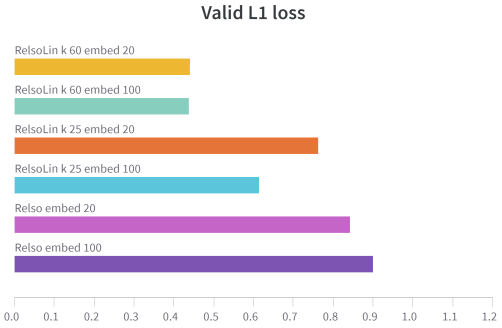
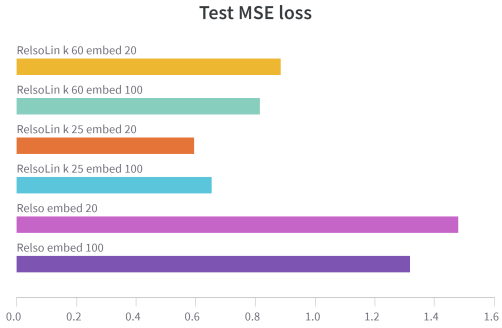


Figure 3: **Cross-entropy Loss on TAPE dataset**: Cross-entropy was computed between original input sequences and reconstructed output sequences with intent to use it as reconstruction loss.

Figure 4: **Pearson correlation coefficient on TAPE dataset**: Pearson correlation coefficient was computed between target fitness values and predicted fitness values.







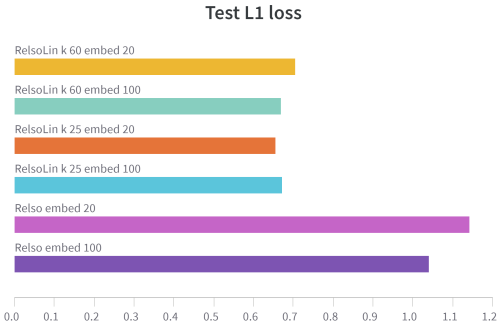


Figure 5: **MSE Loss on TAPE dataset**: MSE Loss was computed between target fitness values and predicted fitness values.

Figure 6: **L1 Loss on TAPE dataset**: L1 Loss was computed between target fitness values and predicted fitness values.

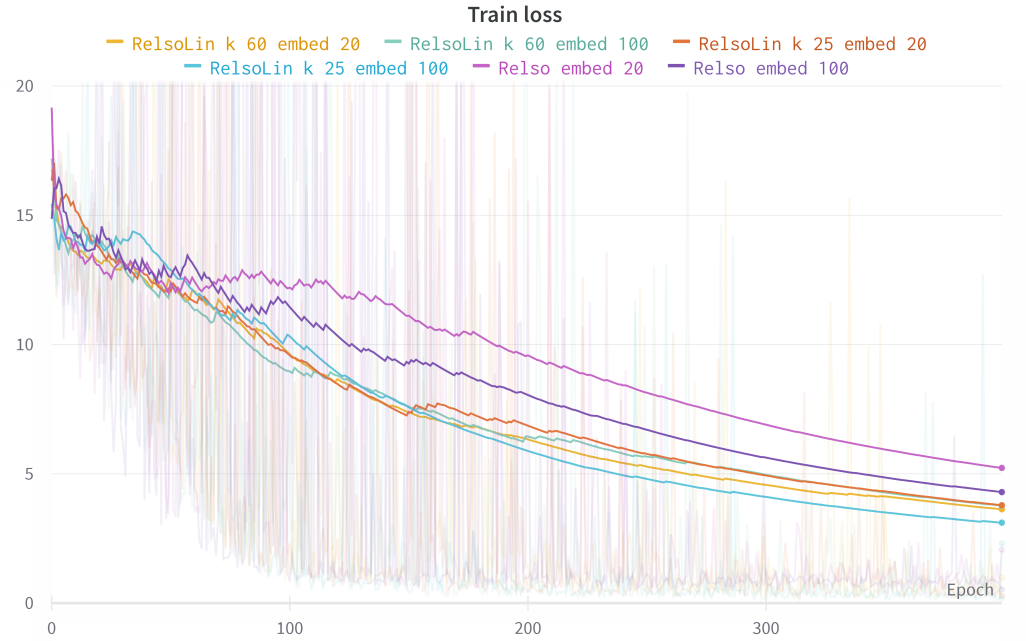


Figure 7: **Training loss**

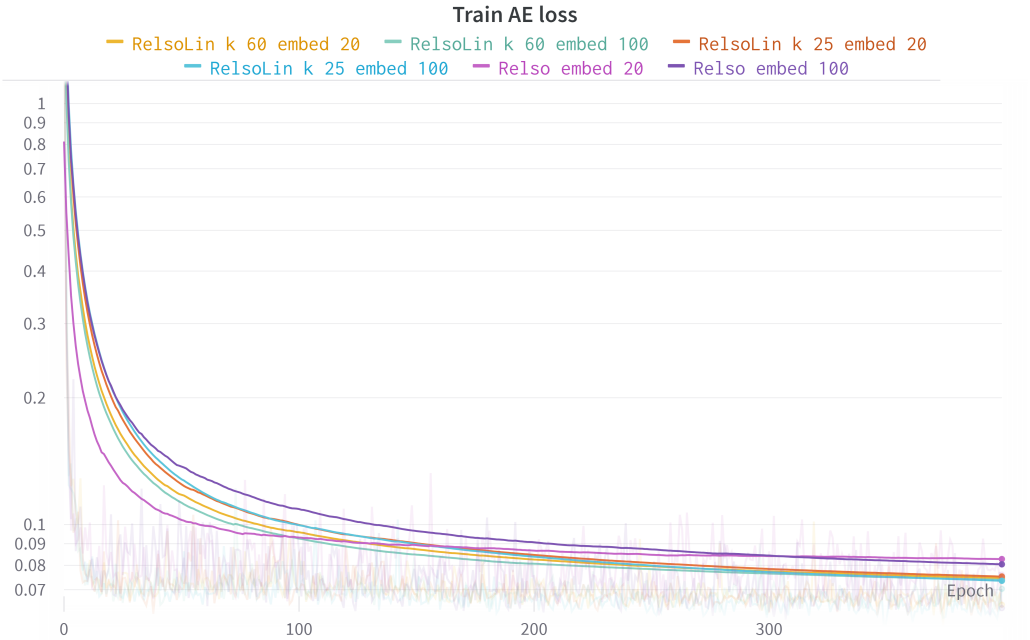


Figure 8: **Training AE loss**

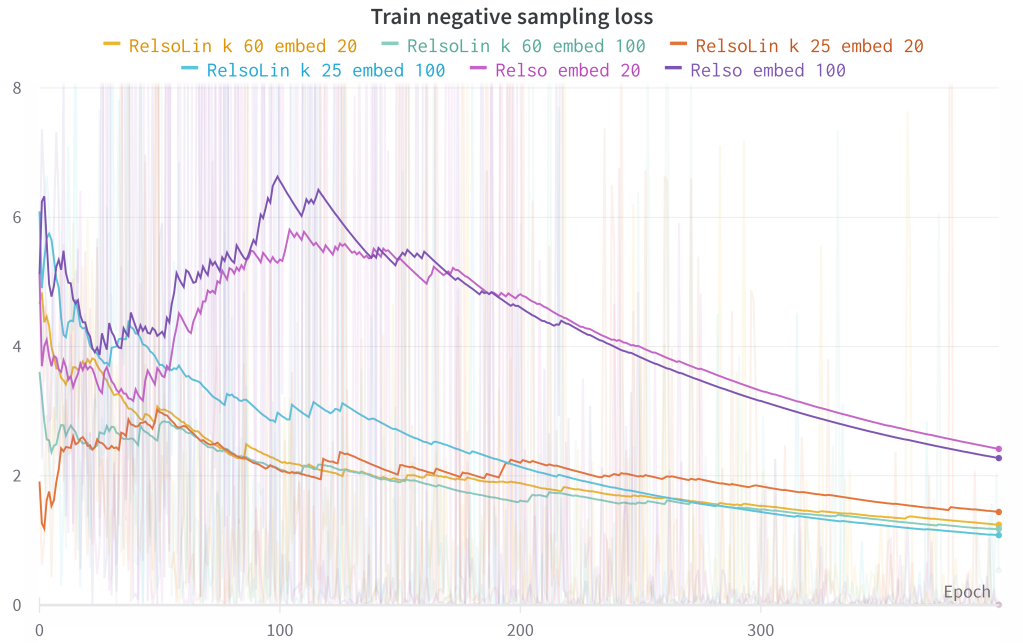


Figure 9: **Training negative sampling loss**: measures the dissimilarity between the original input sequences and the reconstructed output. We can observe that LinReLSO models consistently obtain lower loss values, which indicates a better reconstruction ability.

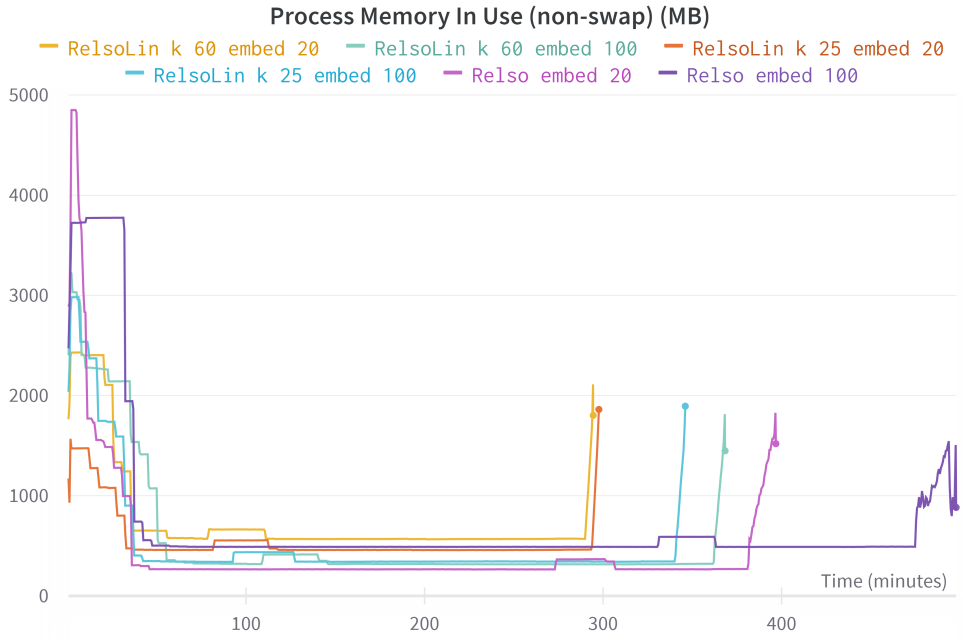


Figure 10: **Process memory in use (MB)**

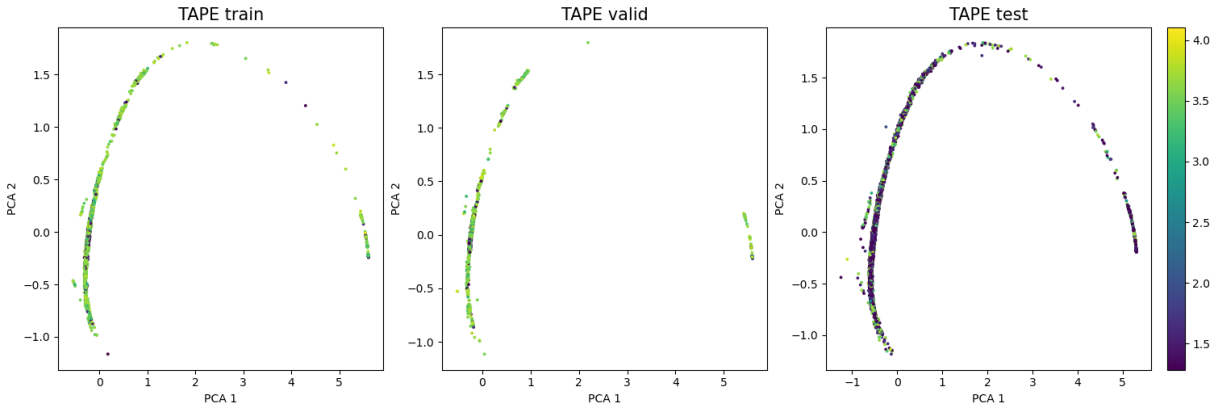


Figure 11: Embeddings, ReLSO with $embedding\_dim = 20$

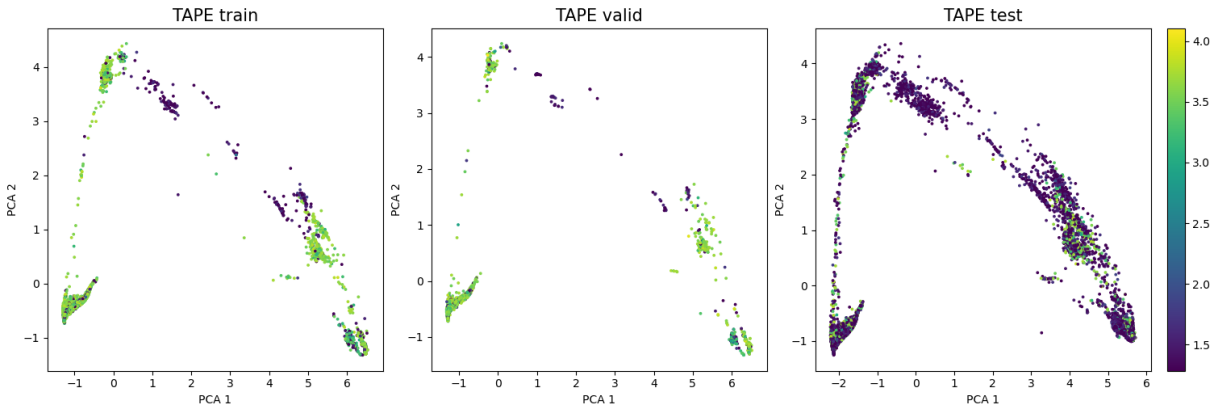


Figure 12: Embeddings, ReLSO with $embedding\_dim = 100$

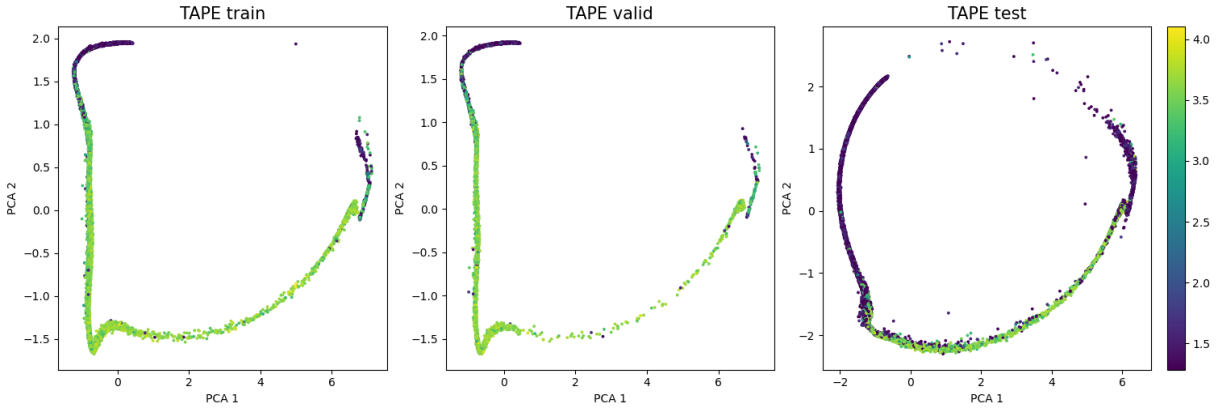


Figure 13: Embeddings, ReLSO with Linformer with $embedding\_dim = 20$ and $k = 25$

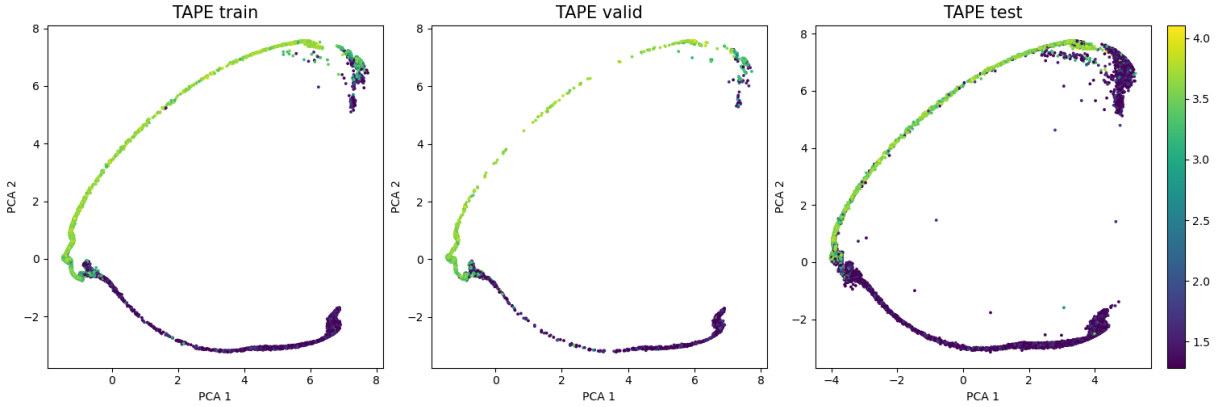


Figure 14: Embeddings, ReLSO with Linformer with $embedding\_dim = 100$ and $k = 25$

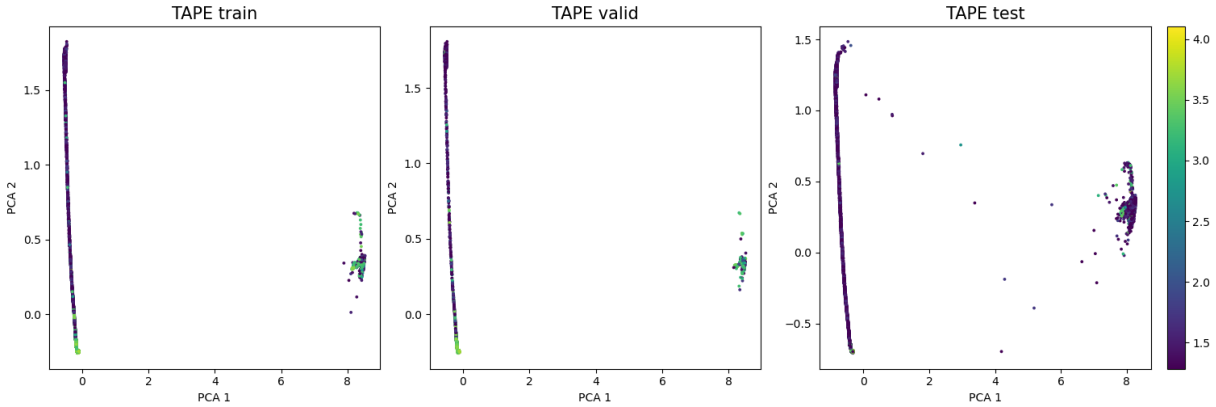


Figure 15: Embeddings, ReLSO with Linformer with $embedding\_dim = 20$ and $k = 60$

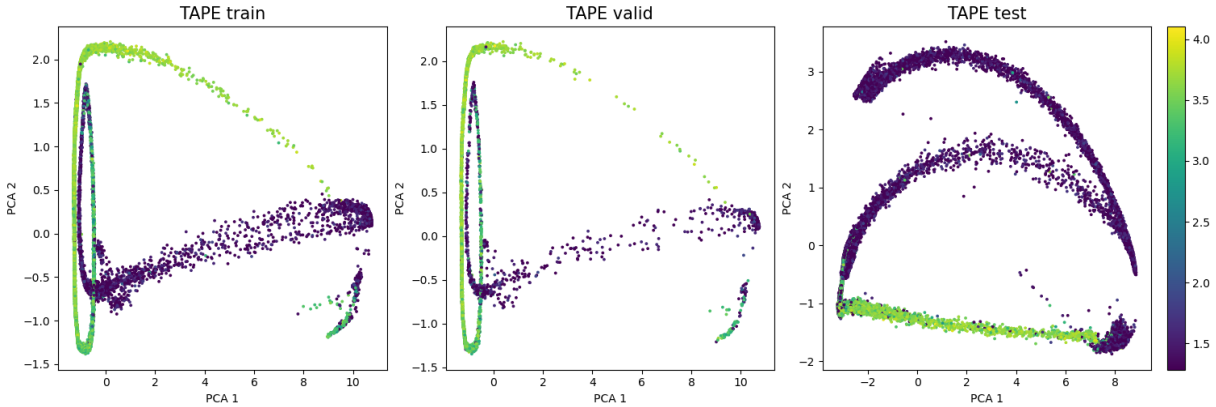


Figure 16: Embeddings, ReLSO with Linformer with $embedding\_dim = 100$ and $k = 60$