Comparing different autoencoder architectures in order to predict nanobodies sequences from their 3D structure

Adi Avni, Iris Schodl, Shlomit Gila, Avishai Elmakies, Noam Ben Sason

Introduction

Nanobodies are proteins that are part of the immune system. Nanobodies are single domain antibodies which were first found in camelids. They are capable of specific binding to antigens and have potential in the treatment of Covid - 19, Alzheimer's disease, Cancer and more.
The antibody is a complex of four proteins , It consists of two light chains and two heavy chains. There are two parts of an antibody , Fc which is a preserved and fixed and two Fabs which have a preserved part and a variable part. We will be interested in the variable parts in the Fabs, it has three complementarity determining regions - CDR1, CDR2, CDR3 which form three variable loops. Those loops are an important part of the antigen and are also the most challenging parts to model since they vary from one antibody to another.

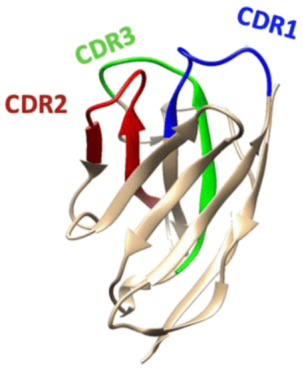


Figure 1: A nanobody structure. Marked are three variable regions of the antibody - CDR1, CDR2, CDR3.

Due to the huge success of AlphaFold2, neural networks have become a common tool for modeling proteins. In exercise 4, we built a neural network that predicts the 3D structure of a given nanobody sequence.

Autoencoder is a type of artificial neural network which uses unsupervised learning (unlabeled data). It is composed of two submodels - an encoder and a decoder. The encoder usually takes data from a specific sample space and tries to represent it in a different space. The decoder attempts to recreate the data provided by the encoder. AEs are used for learning a compressed representation of raw data and applied in many fields such as - facial recognition, acquiring the meaning of words, feature detection and more.

In this project, we decided to try an autoencoder based architecture for the task of Protein sequence design\prediction from a 3D structure of that protein. We also experimented with different model designs and parameters of the AE to see which will give better results.
After the training, we will use the encoder as the design network.

Methods & Materials

Data

Our data for the project was taken from the data given to us in Ex4 and contains 1974 pdb files. The input structures in the pdb files are **after structural alignment** to a reference nanobody, this enables us to be invariant to transformations. This is possible because nanobodies have the same structural fold and differ mainly in their specificity determining regions (CDR loops).
Since nanobodies can come in different lengths, we have to find a way to feed them all to our network. To overcome this issue, we used the same solution offered to us in the exercise - a constant nanobody length (140) and all shorter sequences were padded with a special amino acid "-". The output matrix was also padded with zeros.

Data preprocessing

We represented each protein structure as a coordinates table, in a similar way to Ex4.
The table is of size 140 X 15, where rows represent the amino acids and columns represent the x,y,z coordinates of backbone atoms (N, $C\alpha$, C, O, C_β). For each atom there is a (x,y,z) coordinates.

		N			C_α			C			O			C_β	
	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
aa1															
aa2															
…															
aa140															

Network architecture

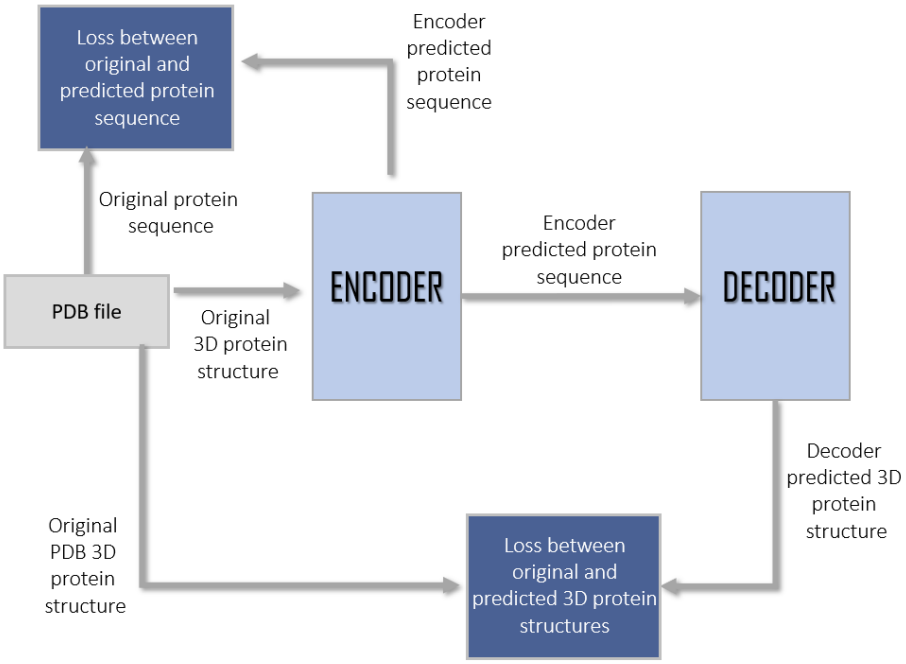


Figure 2: Our autoencoder network architecture.

As mentioned, our model includes two sub-model - the encoder and the decoder, and the backpropagation process is performed based on two weighted losses - a MSE loss between the original 3D structure and the predicted 3D structure (the output of the decoder), and a Categorical Cross entropy loss between the original sequence and the predicted sequence (the output of the encoder).

Categorical Cross Entropy is a Loss function mostly used for categorical prediction, it can be used with binary classification and multi class classification. We consider our classes as a probability over some space and try to predict that probability using our model with the softmax activation function and then use cross entropy on those distributions for discrete probabilities

cross entropy is defined as $H(p,q) = \sum_x p(x)\log(q(x))$ this is a known function that compares the similarity of two distributions. In this case we compare the probability over the amino acids predicted by our model against the real sequence.

<u>Encoder Architectures</u>
We built three different network architectures and compared them with different versions of decoder architectures, in order to find the combination in which our network will work best.
The first Encoder architecture is the "vanilla" one. It's architecture is similar to the decoder, but without the batchnorm (we thought it might work better) and with leakyRelu instead of Relu as activation.

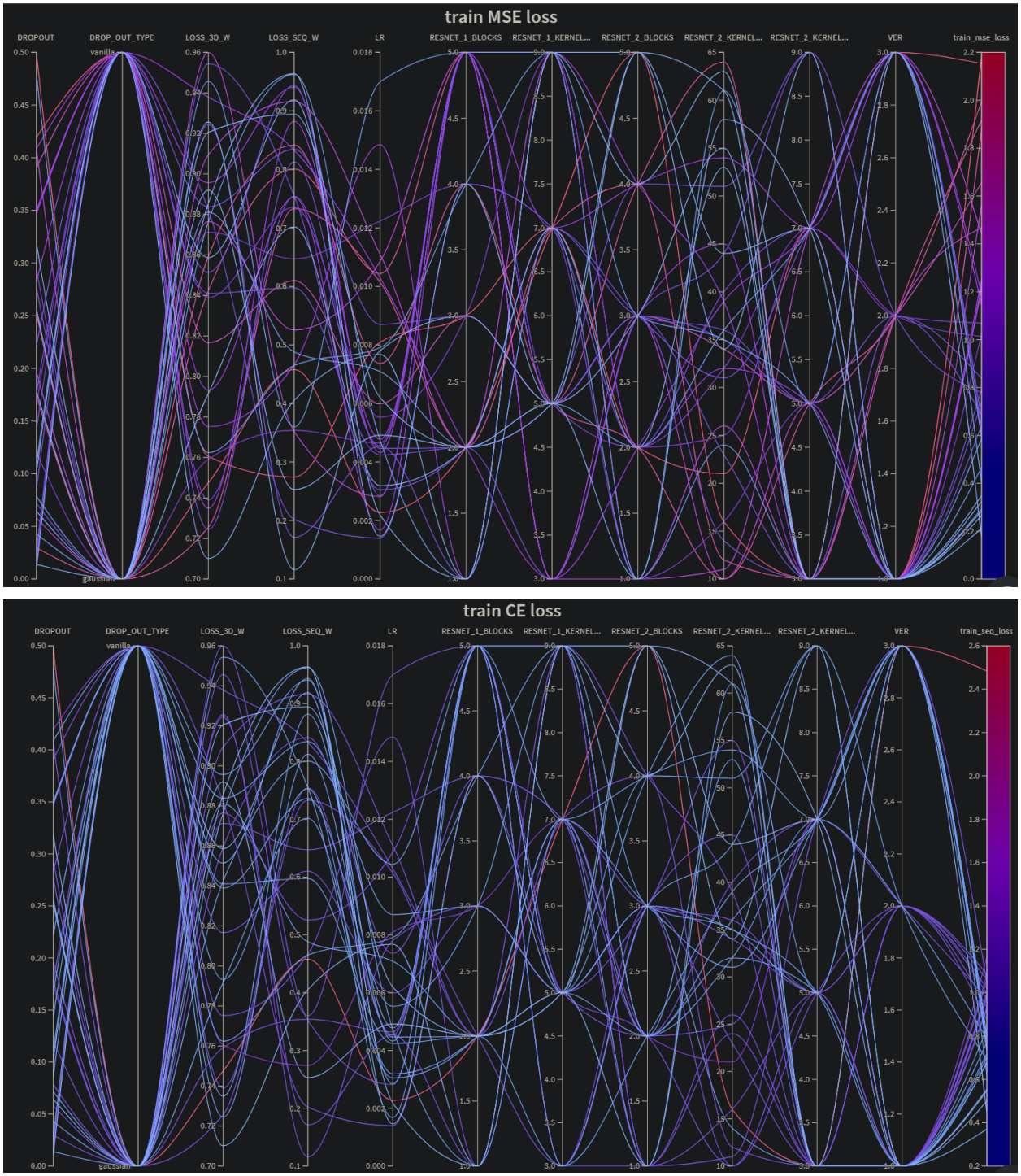
The second Encoder architecture is with attention. Since the structure and sequence of the protein are affected by all of the atoms in the 3D model we thought that maybe giving the model more context would work better for the task we chose, we decided to change the first 1D convolution we had in the original model with a simple layer of attention that will weight the amino acids based on relevance and proximity to other amino acids.
The second Encoder architecture is with multi headed attention. Since attention seemed to work well within our limited testing in the beginning we decided to try the more powerful MultiHeadedAttention Layer instead of the attention Layer. Thus giving the encoder an architecture which is a combination of the transformer and our model from ex4. We also added BatchNorm again to this architecture because it seems to work well with MHA.

In all the versions of the encoder and decoder we tried different parameters (the names are listed as they appear in the code):

- leakyAlpha - the leaky relu parameter
- LOSS_3D_W - the weight of the MSE loss in the train loss
- LOSS_SEQ_W - the weight of the Cross entropy loss in the train loss
- CLIP
- DROP_OUT_TYPE - regular dropout vs a gaussian dropout (adds Gaussian noise)
- VER - the version of the encoder: normal, attention and multi headed attention.

We trained different AE network architectures with different combinations of the encoders and decoders, hoping to find the combination that minimizes the loss.
We trained the different networks separately and observed the changes in the network and whether the change made the network better or worse for our purposes.

Hyper- parameters selection
For the hyper parameters selections we used the tool 'weights and biases'. The tool allows us to define discrete values/range for each hyper parameter we wish to examine. Then, it can train a model based on a set on parameters it randomizes (a set seen in the code as config)



Figures 3a (MSE loss) and 3b (CE loss): The x axis contains all the hyper-parameters that we used the tool 'Weights and biases' to choose. Each line in the graph is a single run on the specific randomize config (hyper parameters set). As a run is more blue, the specified loss is lower, and as the run is more red, the specified loss is bigger.

As we can see in figures 3a, the vanilla version (VER 1) and the multi headed attention version (VER 3) of the model did better on the 3D structure recovery task then the attention version.

(VER 2), since the blue runs are all from those versions. In the sequence prediction task the attention model did better but not much better.

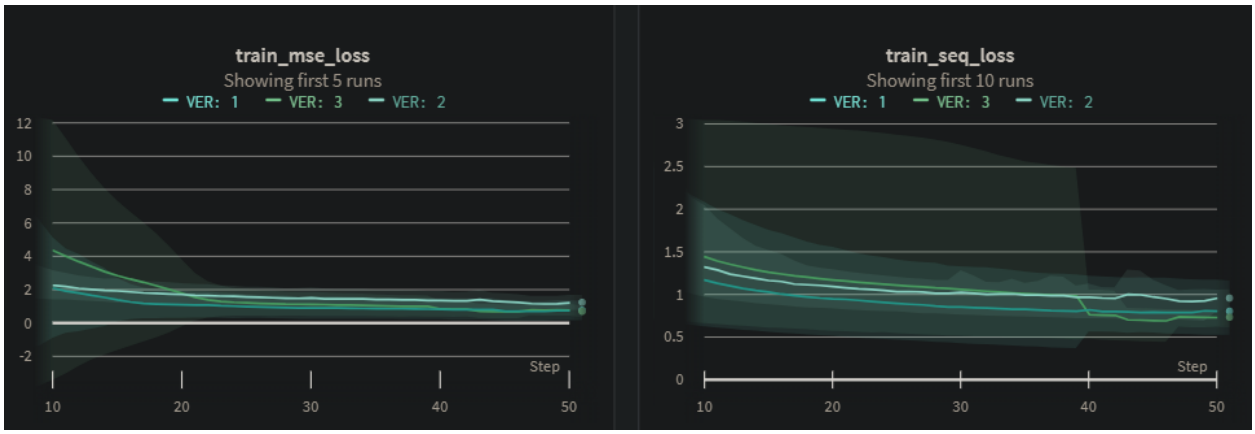


Figure 4: comparison between the mean of the 2 losses on all 3 versions of the model. Version 1 - vanilla model, version 2 - attention model, version 3 - multi headed attention model.

From figure 4 we can infer, that the vanilla version and the multi headed attention version are better than the attention version. We can see that version 3 is a little bit better.

We chose 3 final models, one of each type, to be the models used in the user manual.

	MSE loss	**CE loss**
Chosen vanilla model	0.265028715133667	0.3704842627048493
Chosen attention model	0.7784700989723206	0.6907316446304321
Chosen multi headed attention model	0.17048056423664093	0.4451261758804322

Results

After training all three models with different parameters, with a total of 111 runs using wandb as a tool, we picked the best performing models for each model version we tested. We used those models for predicting sequences and 3D structures of proteins from the test set we were given. Here are our results.

Our goal was to predict a protein sequence from a 3D structure, here are the results of the sequence prediction from our models:

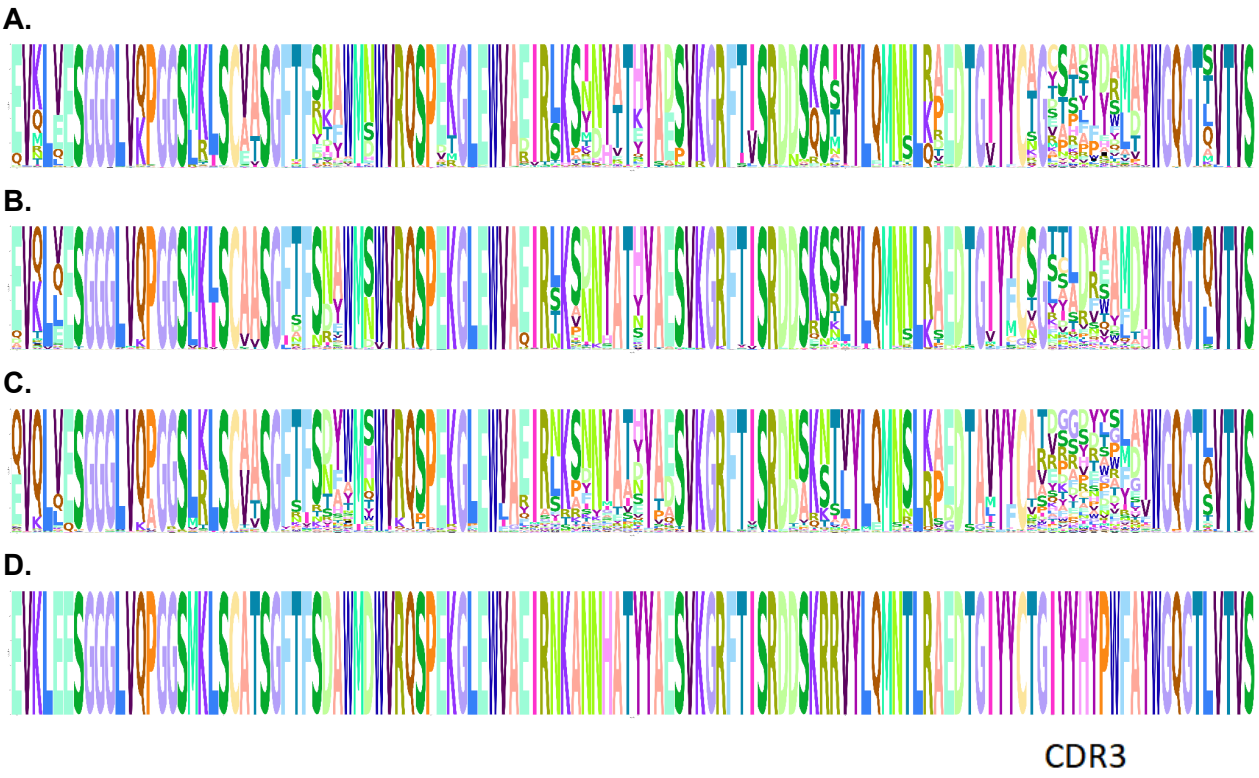


Figure 5: Sequence Logo of predicted sequence of a given structure (PDB code - 1dlf) and the original sequence. Each prediction is from a different model. The x axis represents the different positions of the sequence (from 1 to 140) and the y axis (size of the letter) represents the counts per amino acids character. 2.A - Multi Headed attention model, 2.B - Original model, 2.C - Attention model, 2.D - the original known sequence.

In the plots above, we can see that for all three models, the hardest part to predict was the CDR3 region [indexes 98-109]. This region is known to be the hardest to predict due to the diversity of length and sequence[1].

[1] Higashida, R., & Matsunaga, Y. (2021). Enhanced Conformational Sampling of Nanobody CDR H3 Loop by Generalized Replica-Exchange with Solute Tempering. *Life (Basel, Switzerland)*, *11*(12), 1428. https://doi.org/10.3390/life11121428

Sequence prediction accuracy

Another criterion for validating our results is the accuracy of the predictions for each region. These values are summed in the following table:

Model	cdr1_accuracy	cdr2_accuracy	cdr3_accuracy	Total model accuracy
Vanilla	0.67	0.61	0.42	0.81
Attention	0.60	0.50	0.39	0.74
MultiHeadedAttention	0.66	0.60	0.43	0.81

Figure 6: the accuracy of the predicted sequences compared to the original protein sequences. We can see the accuracy for each model and each CDR.

As we can see, the accuracy for all three models is the highest for cdr1, followed by cdr2, with cdr3 having the lowest accuracy out of all the regions. The vanilla model's and the MultiHeadedAttention model's prediction accuracy was similar and the attention model gave lower accuracies.

One of the outputs of our model is a 3D structure, which it tries to recreate from the input of the encoder (see network architecture). We used MSE along with our CCE (to train our sequence prediction) for this purpose. We can compare the results for both losses on the test set:

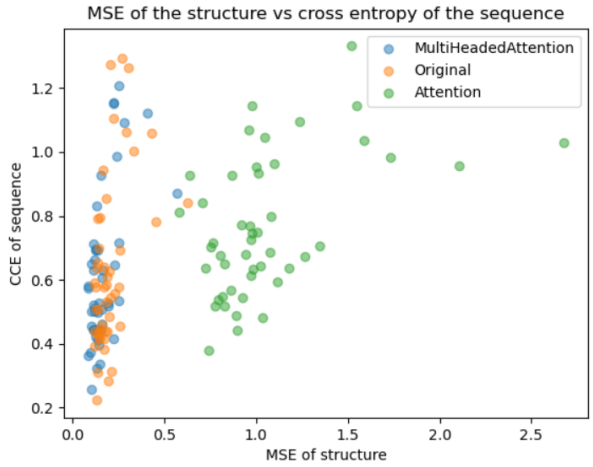


Figure 7: Scatter plot of the MSE of the proteins' structures vs the CCE of their sequences. Values from different models are in different colors. The x axis represents the MSE of the proteins' structures and the y axis represents the CCE of the proteins' sequences. Blue - Multi Headed attention model, Orange - Original model, Green - Attention model.

The plot above shows the MSE (Mean Square Error) of the protein's structure versus the CCE (Categorical Cross Entropy) of the protein's sequence.

We can see that in this sense, the multi headed attention model is very similar to the original model. Furthermore, the Attention model seems to perform worse than the other two models and is more distributed in comparison to them.
As for the original model and the multi headed attention model, it is prominent that the MSE of most samples was lower than their CCE.
Therefore, we can conclude that these two models predicted the proteins' structure better than their sequence.
It is also noticeable that the MSE values showed a lower variation than the CCE values.
This can be explained since our model was trained using more weight on the MSE then the CCE, since we want our designed model to predict the correct 3D structure,and thus recreate the the 3D structure better

Since we have special interest in the CDR's of the nanobody we would like to compare the result of their recreation:
In order to separate the variable part of the antibody we isolated the variable part of the antibody, to test our results and calculate the RMSD. If we calculated the RMSD on the entire antibody, we would get a very low RMSD value, since the constant (structural) part of the antibody doesn't change and therefore would probably be predicted correctly all the time. This will also lead to a low variance in the RMSD values. Since this is less informative, we decided to isolate the variable part of the antibody from the constant part.

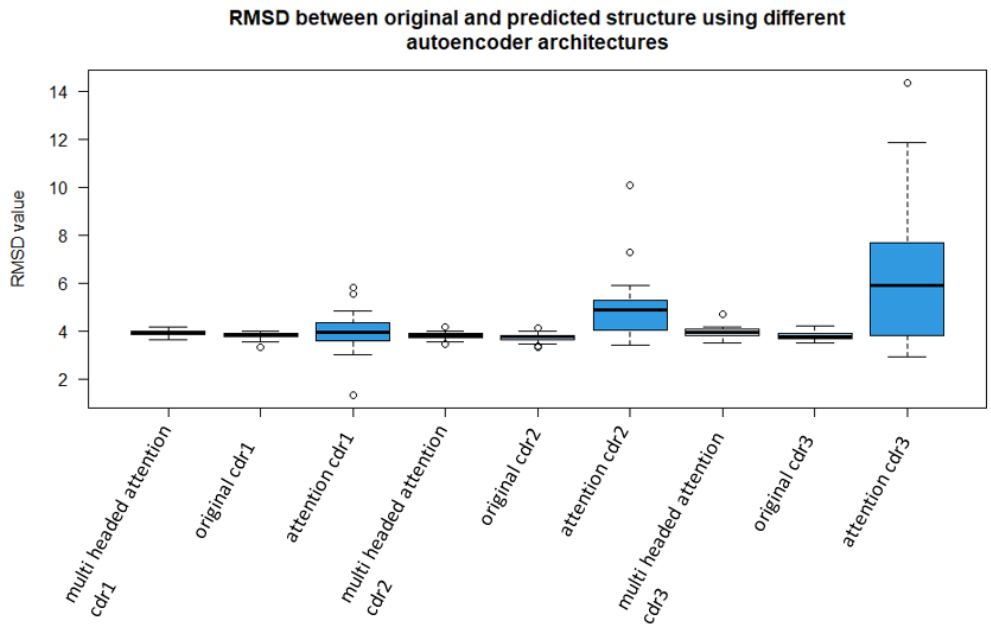


Figure 8: Boxplot of the RMSD between original and predicted structure of the different samples in the test set. The x axis represents the model architecture(original ("vanilla" model), attention, multi headed attention) and the y axis represents the RMSD value.

Observing the plot (figure 8), we can see that generally, the attention model showed significantly higher RMSD values than the other two models. Moreover, the attention model had higher variance.
The original vanilla model and the multi headed attention model performed similarly.

In addition, we chose one protein - 1dlf and aligned the reference protein to predictions of our three models. Results can be seen below.

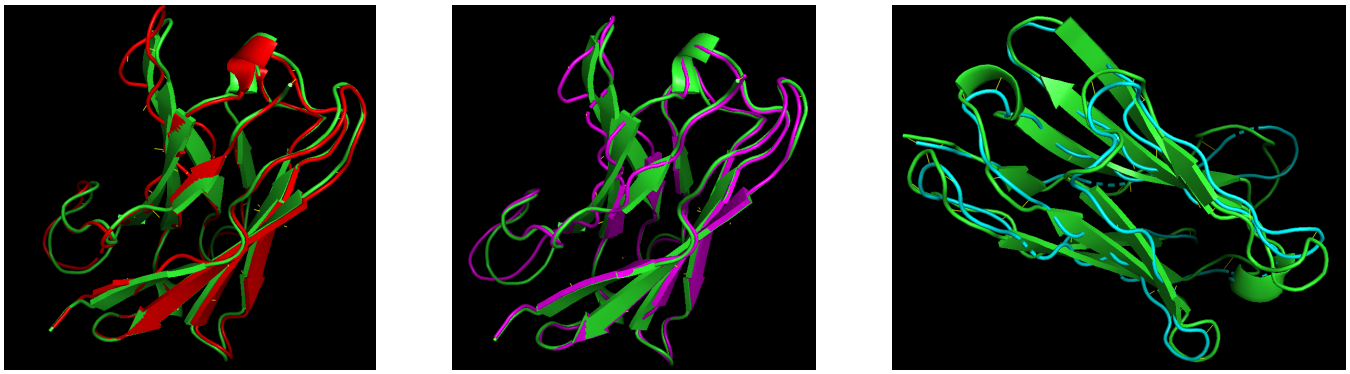


Figure 9: left: In green- 1dlf from the test set, in cyan- the predicted protein according to the attention model. mid: In green- 1dlf from the test set, in pink- the predicted protein according to the multi headed attention model. right: In green- 1dlf from the test set, in red- the predicted protein according to the "original" model.

It can be seen from the alignments that the predictions of all three models are relatively good

Discussion

In this project we created a model for protein sequence design/prediction. We saw that the original and multi headed attention encoders did a better job at reconstructing the 3D structure and predicting the sequence of the structure, while the attention based encoder did much worse on those tasks. This can be explained since both the vanilla (convolution based) version and multi headed attention version seem to learn the "context" of the protein structure much better than the normal attention version which doesn't have any weights learned in it.

limitations

When building the model, we didn’t take physical restraints into consideration such as atoms distances, repeltion between atoms, charges of amino acids...
In addition, our model didn't verify that the two FABs of the antibody were identical and indeed had the exact same sequence of amino acids.
We recommend ensuring that these conditions hold and aim to do so in the future. This can be achieved by adding them to our loss calculation in some way or predicting multiple sequences and choosing the ones that adhere to those constraints.

Conclusions

As we saw, using an AutoEncoder architecture is useful for a lot of applications. We could also conclude that the architecture can give good results for the task of protein design/prediction.
We also saw that our convolution based autoencoder (vanilla model) and multi headed attention based autoencoder managed to predict a sequence (given the structure) with 81% accuracy.
From the found losses we saw that there is a slight advantage to the multi headed attention model, so we would prefer to use the chosen model from this type to the design task.

Figure 10: a camel