Neural Networks and Deep Learning in Science
Final Project- Finding MIMO

Sarah Baalbaki, Tanxin Qiao, Jackie Vo

I. Overview

The use of Neural Networks in science has increased, in various tasks such as image recognition, natural language processing, and medical diagnostics. They have changed the way we approach complex problems, and their impact on scientific discovery and innovation is increasing.

For our project, we decided to approach a new type of problem. As we know, protein sequences in our body have mutations such as substitutions, insertions, deletions.. These mutations affect the protein structure, and as a result the protein function too, leading to impaired biological functions or even pathogenic consequences.

The task we decided to deal with is detecting the avGFP (Aequorea victoria Green Fluorescent Protein) log fluorescence signal levels, which vary based on the mutations found in the protein sequences. AvGFP exhibits strong fluorescence properties that make it a valuable marker for fluorescence microscopy, protein labeling, and biosensing. We care about the relationship between the sequence variations and the resultant fluorescence signal levels since the sequence mutations lead to changes in the structural, chemical, and functional properties of the protein and so, they influence the fluorescence emission characteristics of avGFP, which affects the intensity and duration of the emitted fluorescence signal. So, by determining the mutations in the sequence, we expect to learn and predict the approximate GFP expression level (log fluorescence level) for previously unseen sequences.

We have some signals with 1, 2, 3,.. up until 14 mutations. We split our data such that our training data has either 1, 2, or 3 mutations, and the test data has between 4 to 14 mutations. As a result of this data imbalance, we decided to work with a model called MIMO, which is a Multi-Input Multi-Output model for protein fluorescence signals. This model has multiple subnetworks, but allows us to make multiple independent predictions in a single forward pass, and combine or average out the different predictions to produce the final output. So, each subnetwork will be dealing with different mutations, and we will average the log fluorescence from all the obtained subnetworks to get the resulting log fluorescence of the mutated sequence.

II. Data

The dataset we are using is obtained from TAPE (Tasks Assessing Protein Embedding), and it contains the avGFP protein sequences, their lengths, number of mutations, and corresponding log fluorescence levels.

We split our data as such:

- Train Data: 21,446
- Validation Data: 5,362
- Test Data: 27,217

We also ensure that our training data has either 1, 2, or 3 mutations, and our test data has between 4 to 14, as seen in Figure 1.

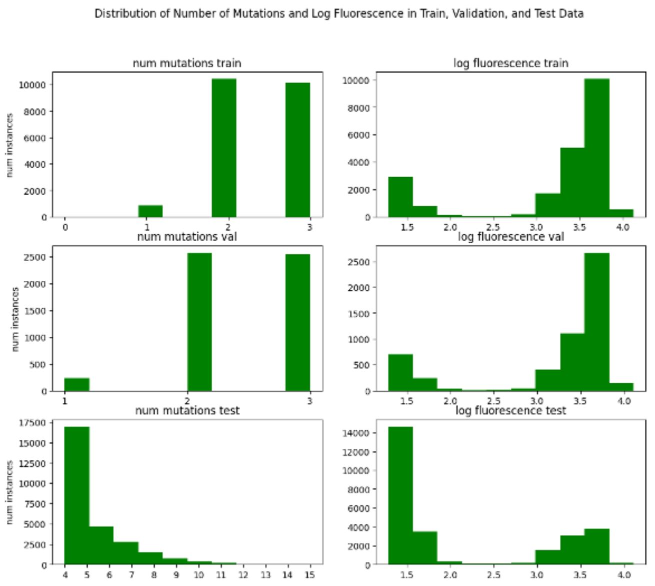

Figure 1: Bar plots showing the number of instances of train, validation, and test data versus the number of mutations and the log fluorescence

As we can see from the figure above, the train and validation data have a log fluorescence distribution which is right-skewed, indicating a higher log fluorescence value over those for the test data with more mutations, where the log fluorescence is left-skewed, indicating lower log fluorescence values.

The protein sequences we have are not of the same length due to mutations (indels), and so we wanted to initially perform a multiple sequence alignment of all the data, then tokenize and encode the sequences. However, we found this to be computationally expensive since our dataset is very large, and we are left with an imbalance length between training, validation, and testing data.

III. Our Approach and Models

To tackle our problem, we had several different approaches. But for all approaches, we preprocessed the sequences in the same way. We used the ESM model tokenizer and embeddings from HuggingFace, ran our sequences through them, and then used the results as input into our model.

For our first approach, we ran an analysis on a shuffled version of the dataset, where we had our training, validation, and testing data having the same numbers of mutations, and having a more uniform log fluorescence distribution, as shown in Figure 2. This model's aim was to serve as a baseline for comparison in reference to other models trained on the skewed data.

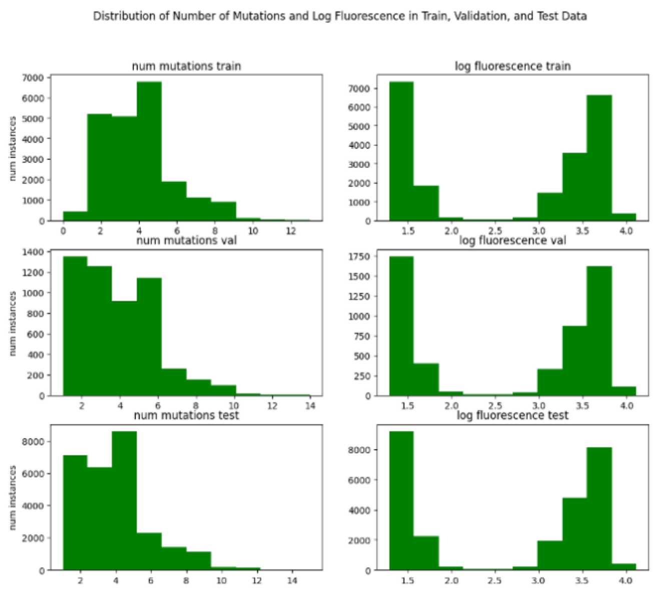

Figure 2: Bar plots showing the number of instances of train, validation, and test data versus the number of mutations and the log fluorescence levels, for a balanced version of the data split.

After that, we started working with our imbalanced dataset. We formed a regression model using the same architecture as the base model, and then made a more complex model using convolutional layers, ReLU, and linear layers. The results are displayed in Table 1.

For the final approach, we implemented MIMO, a model with three subnetworks that takes three inputs and outputs three predictions in a single forward pass. Each subnetwork has a separate input layer and output layer but shares the same middle layers. Each subnetwork was trained on input sequences with a fixed number of mutations, i.e., subnetwork1 only takes input sequences with one mutation, subnetwork2 only takes input sequences with two mutations, and subnetwork3 only takes input sequences with three mutations.

Specifically, the three outputs after the first linear layers will be concatenated and go through the middle layers together and the outputs out of the last middle layer will be split into three portions to go through the three final layers separately (Figure 3). For validation, the same splitting scheme was applied. For testing, each sequence was fed to all the subnetworks and the average of the three outputs would be obtained as the final prediction.

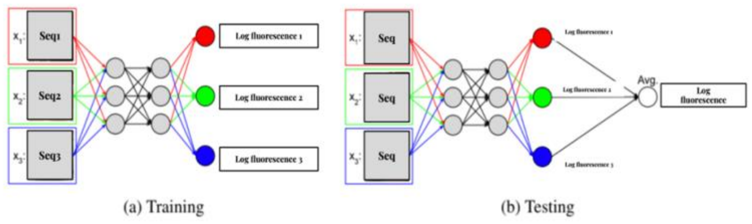

Figure 3: MIMO Model implementation illustration (train & test steps)

Based on the above model structure, a major challenge is that the number of inputs for each subnetwork must be kept the same during training and testing. However, the distribution of sequences with 1, 2, and 3 mutations are fairly skewed (Table 1). In order to address this issue, the simplest way is to crop each subset to the subset that has the minimum size. However, this will leave us with only 877*3=2631 training instances, which would be very limited to test on a test set with 27,217 instances. Therefore, for each epoch we shuffled a different training set that contains the same number of instances for each subnetwork. In other words, the instances with 1 mutation would all be trained on each epoch but the instances with 2 or 3 mutations would vary for each epoch, effectively including most of the training set and increasing the weights of instances reverse proportionally to their distribution.

Mutations/Dataset	**Training**	**Validation**
1	877	237
2	10433	2577
3	10135	2548

Table 1: Number of instances with different number of mutations in the training set and the validation set

Additionally, because of the lack of training instances (2631) per epoch, the model could be stuck on a lock minimum very easily. Therefore, in each epoch we selected 8770 instances from the instances with 2 and 3 mutations respectively and combined with the 877 instances with 1 mutation ten times, to get a more stable training process.

IV. RESULTS

For the models, we modified the learning rates and tried to retain the best performing ones for each model, as shown in Table 2.

Model	Epochs & LR	Train Loss	Validation Loss	Test Loss
1- Base model on balanced data	50 epochs 1e-2	0.656	0.655	1.08
2- Base model on imbalanced data	50 epochs 1e-2	0.4853	0.5050	1.647
	50 epochs 1e-1	0.516	0.434	0.873
3- Improved model on imbalanced data	50 epochs 1e-4	0.428	0.399	1.167
4- MIMO model	30 epochs 1e-3	2.13	2.18	0.96

Table 2: Performance and specifications of different model structures implemented

As we can see from table 2, train and validation losses for the baseline models are relatively lower than that of the MIMO model. However, MIMO performs better on unseen data relative to the baseline models.

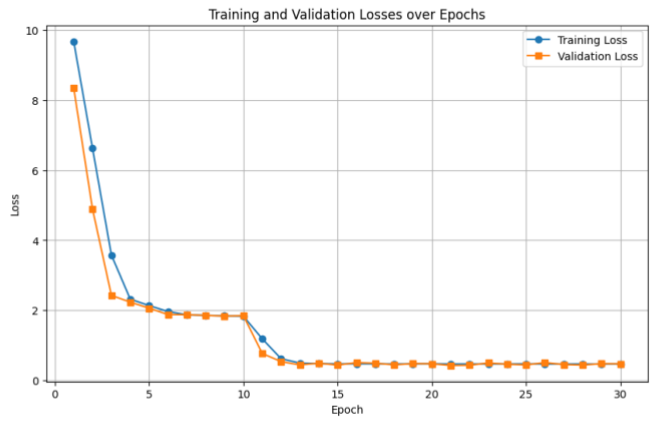

Figure 4: Training and validation loss of MIMO model over epochs (Adam optimizer, StepLR scheduler, learning rate = 1e-4, MSEloss)

Figure 4 is a plot for the training and validation losses of the MIMO model over epochs, which we evaluated using the MSE loss since we have a regression problem. We can see a decreasing trend in these values, but one thing to note is that the model eventually starts to overfit, as we see from the testing results. Thus, it is crucial to know when to stop training our models so that we ensure a balance between training and testing performance.

V. CONCLUSION AND FUTURE WORK

The models with one input and one output, despite the descent training and validation loss, did not perform well on accumulative mutation testing data. This is expected because the models are trained on sequences with a few mutations, and tested on very different ones with more mutations, so this behavior is expected.

We expected to see some improvement using the MIMO model because it takes into consideration a committee in order to make predictions, and so if each model is trained on some mutations, we can aggregate the results to form a more meaningful prediction for the sequence with longer mutations.

However, the MIMO model did not outperform the best base models. What is interesting about the MIMO model's performance is that although the training and validation loss keeps dropping (Figure 4), the test loss was the lowest when the training loss and validation loss were around 2.13 and 2.18. This indicates that the decreasing loss after that may be due to overfitting to the training data but also shows a positive signal of the generalizability of MIMO model since the valid and training loss were a lot higher than test loss.

Overall, the MIMO model has the potential to provide a more reliable result on out-of-distribution prediction tasks but there is certain limitation on the number of input instances to each subnetwork. The other challenge is that it is hard to determine when the model starts to overfit and loses the generalizability to new instances.

REFERENCES

[1] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M. Dai, and Dustin Tran. "Training independent subnetworks for robust prediction." CoRR, abs/2010.06610, 2020. [Online]. Available: https://arxiv.org/abs/2010.06610

[2] Rao, Roshan, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. "Evaluating Protein Transfer Learning with TAPE." (2019).