MLB S25 Hackathon Guide:

ML-Guided Protein Engineering

Introduction

Protein mutations refer to the changes in the amino acid sequence of a protein, typically due to genetic mutations in the corresponding DNA sequence. These mutations can lead to changes in protein structure, stability, or function. Understanding how mutations affect protein function is a fundamental challenge in the field of computational biology, with a wide range of applications in biology and healthcare (e.g., protein engineering, drug design, and synthetic biology). Although experimental biologists can measure the mutational effects of mutants for a given protein in the wet lab, such efforts are costly and time-consuming, limiting the total number of mutants to be tested.

The goal of this hackathon is to develop an accurate computational method to predict the functional impact of protein mutations using machine learning. We will solely focus on single-site substitutions (i.e., only one residue in the mutant protein sequence differs from its counterpart in the wild-type protein sequence). As there are 20 different types of amino acids in total, the number of all possible single-site mutants for a protein with length L is $19L$.

You will experience a simulated scenario: after acquiring limited experimental measurements from the wet lab, you will need to construct an effective mutation effect predictor to make function predictions for all unmeasured mutants and then recommend mutants with the most improved function. Each team will start with a small fraction (about 10 percent) of the amino acid positions and will have the opportunity to query additional data points in a structured manner, simulating an active learning setting.

The final objective is to use the trained models to recommend the top 10 beneficial mutations that maximize protein fitness. The performance of each team's model will also be evaluated based on the Spearman correlation between fitness predictions and ground truth fitness measurements on the held-out test data. Additionally, teams will submit a detailed project report outlining their methodology, training strategy, and decision-making process. On top of these, we will be using a leaderboard released on Gradescope to rank the model performances; bonus points will be awarded to the leading teams.

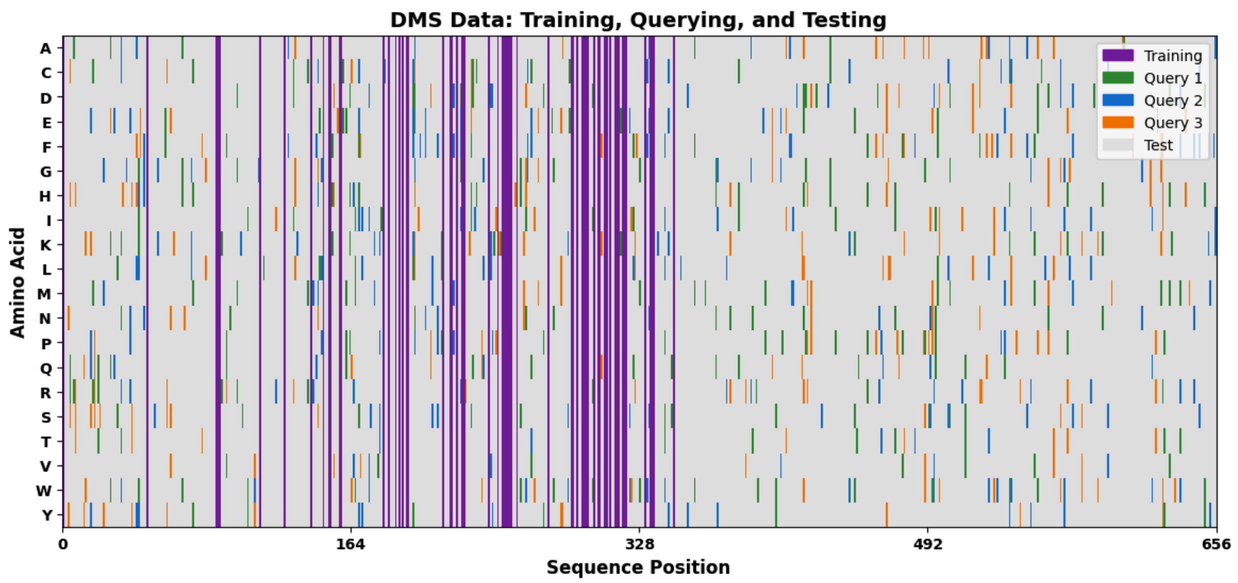
Hackathon details

Overview

At the start of the hackathon, each group will be provided with the experimentally measured fitness data of all mutants from about 10% of the amino acid positions as their initial dataset. All data in this hackathon will consist exclusively of single-point mutations, meaning you do not need to account for higher-order mutations during training or testing.

Beyond the initial dataset, each group will have three opportunities to query random data points (no more than 100 data points each time) from the set of remaining untested mutants. These queries can be made at any time, whether all at once or spaced out, but each group is strictly limited to three queries in total. Data sharing between groups is strictly prohibited, and any unauthorized use of another group's queried data will be detected and addressed accordingly.

Once training is complete, each group will enter the final evaluation round, where they must use their trained model to predict and recommend the top 10 single-site mutants expected to yield the highest fitness scores. These predictions will be evaluated based on accuracy and effectiveness. To facilitate transparency and competition, a leaderboard on Gradescope will be available during the hackathon, allowing you to track their model's performance in comparison to other groups. The goal is to develop the most effective model for identifying beneficial mutations while strategically utilizing the limited query budget.



From the figure above, you can intuitively see how the whole dataset is partitioned. You can distribute your queries in any configuration you prefer. Finally, your model will be evaluated on all test positions, which include all data not used in training – note that this includes the data you queried during the active learning cycle.

Task description

The goal of protein fitness prediction is to develop a machine learning (ML) model that can accurately estimate the functional effect of amino acid mutations in a given protein. Specifically, the model **takes a protein sequence as input and outputs a numerical fitness score** (ranging between 0 and 1), which represents how well the mutated protein performs its biological function. The predicted fitness values should strongly correlate with experimentally measured fitness scores.

What You Need to Do

1. Download the starting data
    - README.txt: A text file containing file descriptions.
    - sequence.fasta: A fasta file that contains the wildtype sequence of the target protein.
    - train.csv: A csv file that contains the initial training data.
    - test.csv: A csv file that contains mutants in the test set.
2. Understand Data format
   Each row in the dataset contains:
    - First character: Wild-type amino acid (e.g., M)
    - Number: 0-based position in the sequence (e.g., 0, 5, 10)
    - Last character: Mutated amino acid (e.g., A, C, T, V)

   A maximum of 100 mutations per query cycle is allowed. Once submitted, you will receive the corresponding DMS data for the queried mutations, which can then be used to continue training your model.

   ```
   M0A
   M0C
   M5T
   M10V
   ...
   ```
3. Understand the Model Input and Output
    - The input to the model is a protein sequence, typically represented as a string of amino acids (e.g., "MKTLLVLAVF…").
    - The output is a predicted fitness score (ranging between 0 and 1), which is a numerical value indicating the effectiveness or stability of the protein mutant. A higher value would indicate a better function for the input protein mutant.
4. Train the Model on Labeled Fitness Data
    - Students will be provided with a dataset of mutant proteins and their experimentally measured fitness scores (ground truth), this is also called the deep mutational screening data (abbrev: DMS), for more details please read the background section. (below is an example of the format of the DMS data we will be providing)

mutant	DMS_score
M0A	0.073
V1D	0.688
K59L	0.236

- The task is to train an ML model that learns the relationship between protein sequences and fitness scores. Your model should take in a protein sequence as an input and output a predicted fitness score. In the background section, we have provided a few approaches you can take but you are also welcome to explore other approaches.

5. Evaluate Model Performance
    - You can perform k-fold cross-validation on labeled fitness data first to test the performance of your trained models.
    - The primary metric for success is Spearman's correlation between the predicted fitness scores and the actual fitness scores.
    - A high Spearman correlation (close to 1.0) means that the model ranks the protein mutants correctly in terms of fitness, even if the exact predicted values are not perfect.
6. Generalization to Unseen Mutants
    - The model will be tested on a separate set of unseen protein mutations, meaning that it must generalize well beyond the training data.
    - This requires learning meaningful sequence-fitness relationships rather than memorizing the training data.
7. Download the skeleton code: We have provided some skeleton code that you can use to get started. The purpose of this code is to provide more clarity on the task we are trying to solve & how to get started, however, the model & architecture might not be the best and you are strongly encouraged to explore other models.

Submission details

On Gradescope, we will create 3 separate assignments for you to submit your queries, your final prediction results, and to check the format of your submissions:

1. **Hackathon-FormatCheck**: Ensuring correct submission format. This gradescope assignment will take in your **query.txt**, **GroupName.txt**, and **APIKey.txt**, and return a test result indicating whether your file has the correct format. This can help prevent you from wasting a query opportunity due to wrong formatting.

    - query.txt: The text file containing no more than 100 query mutants. Please make sure that each line only contains one mutant.

    - GroupName.txt: The one-line text file containing your group number.

- APIKey.txt: The one-line text file containing the api key specific to your group. **Please do not share this file with people outside your group.**

We will provide GroupName.txt, and APIKey.txt to each group through email.

2. **Hackathon-ActiveLearning**: You can query three times, each time requesting up to 100 mutants that are not included in the training data (train.csv), by submitting file query.txt to this assignment. We strongly recommend submitting your query.txt to the format check assignment first to ensure you have the right format. For fairness, you will not receive extra query chances, even if your submission is unsuccessful because of format issues. The assignment will return to you the corresponding DMS data for the query points you requested through autograder.

   - query.txt: The text file containing no more than 100 query mutants. Please make sure that each line only contains one mutant.
   - GroupName.txt: The one-line text file containing your group number.
   - APIKey.txt: The one-line text file containing the API key specific to your group. **Please do not share this file with people outside your group.**

3. **Leaderboard**: Final model predictions, submit test predictions.csv to this assignment and the leaderboard will be updated to reflect your test set performance and how it compares to the rest of the class. Your submission file must follow the following format:

   - predictions.csv: A csv file containing your predictions for all mutants in the test set (i.e., 11,324 mutants), regardless of whether they have been queried before or not. This csv file will need to have two columns: **mutant**, and **DMS_score_predicted**. The test set is provided in test.csv.

     a. mutant: Mutation identifier in the format M#A, where M is the wildtype amino acid, # is the 0-indexed position, and A is the mutated amino acid.

     b. DMS_score_predicted: A scalar value prediction of the mutation's fitness.

mutant	DMS_score_predicted
M0A	0.073
V1D	0.688
K59L	0.236

   - top10.txt: The text file containing your selected top 10 mutants in terms of predicted DMS scores. Please make sure that each line only contains one mutant. Please note that only mutants from the test set will be accepted. Please do not recommend mutants contained in the original training set (train.csv).

- GroupName.txt: The one-line text file containing your group number.
- APIKey.txt: The one-line text file containing the api key specific to your group. **Please do not share this file with people outside your group.**

Suggested workflow

To help you effectively manage your time and tasks during this hackathon, please follow the suggested workflow below:

- Familiarize yourself with the provided initial dataset (train.csv) and read the background section in the project doc.
- Set up your computing environment and experiment with the provided skeleton code.
- Submit your first query of up to 100 mutations (ensure query.txt format correctness via Gradescope's format check).
- Incorporate queried data into your training dataset once received.
- Begin training and validating your model, focusing on baseline model selection.
- Conduct further analysis and strategically submit your second query of up to 100 mutations.
- Update your training set with new data.
- Optimize your model: hyperparameter tuning, cross-validation, and initial performance assessment using Spearman correlation.
- Make your final query submission (up to 100 mutations).
- Integrate final query results into your dataset. Perform comprehensive model refinement and final hyperparameter tuning.
- Select your top 10 mutation predictions based on the final model.
- Submit predictions.csv and top10.txt via Gradescope's leaderboard assignment once your model is finalized, make sure to check your format using the “FormatCheck” assignment
- Prepare and finalize your detailed project report documenting your strategy, model architecture, training procedure, and results.
- Submit the final project report.

Timeline

Here are some of the key deadlines

Date	Event	Description
02/27	Hackathon release	Dataset and problem statement are available.

03/24	First leaderboard submission (optional, non-graded)	We will run the evaluation on all submissions that are submitted by 11:59 PM on this day and release the results to you. While this submission is optional and will not be graded, we strongly encourage you to submit your model's predictions to test the prediction performance of your developed model at this point.
03/28	Second leaderboard submission (optional, non-graded)	We will run the evaluation on all submissions that are submitted by 11:59 PM on this day and release the results to you. While this submission is optional and will not be graded, we strongly encourage you to submit your model's predictions to test the prediction performance of your developed model at this point.
04/02	Final submission: Prediction list (required & graded)	Submit final **predictions**. Submit your final prediction. We will refresh the leaderboard for the last time after the submission deadline and calculate your final score.
04/04	Final submission: Final report (required & graded)	Submit **the final report PDF** on canvas

Evaluation

The evaluation of each group's submission will be based on four key components:

- Model Performance on the Test Set (Spearman Correlation)
- Effectiveness of the Top 10 Mutation Predictions (Recall-Based Evaluation)
- Final Project Report
- Leaderboard Performance (Bonus Points)

Please note that we will only run the autograder for Spearman Correlation, Recall-Based Evaluation, Bonus Points on March 28th, April 1st, and April 4th. The runs on March 28th and April 1st are mock runs, giving you two chances to test your model before final submission. Therefore, submissions before March 28t and April 1st are not mandatory and the results won't be counted towards your final score. Your final score will be based on the results of the run after April 4th (the deadline for this hackathon).

Model Performance on the Test Set (Spearman Correlation)

Each group's model will be evaluated on the test DMS dataset that was not part of their training set. We will calculate the Spearman correlation between the model's predicted mutation effects and the actual fitness values from the test set. The Spearman correlation coefficient (ρ) measures the rank correlation between the model's predicted scores and the true fitness scores. It is defined as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where di represents the difference between the ranks of corresponding values, and n is the number of data points. Spearman's correlation evaluates how well the ranking of predicted scores aligns with the ranking of true fitness scores:

• ρ = 1 Perfect positive correlation: the model's rankings exactly match the ground truth.

• ρ = 0 No correlation: the model's predictions do not follow any meaningful ranking pattern.

• ρ = −1 Perfect negative correlation: the model's rankings are completely opposite to the true rankings.

A higher Spearman correlation means the model correctly ranks mutations based on fitness, indicating a better ability to generalize mutation effects to unseen data. Your models will be scored based on how much their predictions match the ground truth. The score for this section is determined by the Spearman correlation:

Spearman Correlation (Test Set)	**Score Percentage**
≥ 0.6	110% (Full Credit +10% Bonus)
0.55 - 0.6	100% (Full Credit)
0.5 - 0.55	90%
0.45 - 0.5	80%
0.4 - 0.45	70%
0.35 - 0.4	60%
0.3 - 0.35	50%
0.25 - 0.3	40%
0.2 - 0.25	30%

< 0.2	0% (No Credit)

Note that the 10% bonus is only for this section of your hackathon grade, which is equivalent to 3% of the whole hackathon project grade.

Effectiveness of Top 10 Mutation Predictions (Recall-Based Evaluation)

In this hackathon, **30% of the total score** will be based on the quality of the top 10 mutation recommendations provided by each participant's model. Each mutation recommendation contributes **3%** to the overall score. To evaluate these recommendations, we will compare their actual experimental fitness scores against all possible mutations for that protein.

For each recommended mutation, we will check whether its actual fitness score falls within the **top X% of all mutations** for that specific protein. Depending on where the mutation ranks within the global distribution of fitness scores, it will be assigned a score based on predefined brackets. The closer a mutation is to the top-performing mutations, the higher the score awarded. A breakdown table will be provided to show how each recommended mutation is scored based on its percentile rank in the dataset.

The final score for this section will be the sum of the scores across all 10 recommended mutations, ensuring that participants are rewarded for identifying the most functionally beneficial mutations. The maximum score that a student can achieve for this section is 30% of the overall hackathon grade.

Top Percentage Recalled	**Points Awarded**
1%	3 (Full Credit)
2%	2.5
5%	2
10%	1.5
20%	1
50%	0.5
100%	0 (No Credit)

Hackathon Report

Each group must submit a **detailed project report** in **PDF format** to **Canvas** by **03/31** through canvas. The report should document the methodology, model development, and evaluation process in a structured and thorough manner. Your report should include the following key components:

- **Querying Strategy**: Explain the rationale behind your three query selections, how you determined which positions to query and why you distributed the data points the way you did, and the reasoning behind your choices. Discuss any adaptive strategies used and how your queries contributed to model improvement.
- **Model Architecture and Justification**: Describe your model architecture, including the selection process, the rationale for choosing specific architectures (e.g., neural networks, regressions, or fine-tuned pre-trained models), and any modifications made to suit the task. Provide sufficient details for reproducibility, such as the number of layers, activation functions, and loss functions used.
- **Training Details**: Outline your training process, including data preprocessing, feature engineering, hyperparameter tuning, and optimization strategies. Specify how the dataset was split for training and validation, how you addressed overfitting, and the key challenges faced during training.
- **Model Performance Analysis**: In addition to the leaderboard evaluation, discuss any other performance metrics used to assess your model's effectiveness (e.g., Spearman correlation, RMSE, ranking performance). Provide visualizations or tables that illustrate key trends and insights from your model's predictions if needed.
- **Reflections and Challenges**: Reflect on the results, challenges encountered, and potential improvements. Discuss why your model performed well (or not) and any limitations in your approach. If applicable, explain the impact of different design choices, such as data selection, feature engineering, or architectural changes.
- **Code submission:** For reproducibility, please include a github link in your PDF report (or a view-only cloud drive link if you do not wish to make your github repo public). Please also include a README file on how to run your code. Note that you only need to include your source code and do not need to upload your model checkpoints or external data.

Please note that your submission will be evaluated based on effort and documentation rather than solely on model performance. We encourage a thorough and well-documented report that showcases your iterative improvement process, justification for choices, and reflections on challenges faced etc.

There is no page limit, but please make sure you concisely provide sufficient details. In general, we are expecting a 2-4 page report. Be sure to clearly communicate your methodology, results, and insights while providing sufficient technical details for reproducibility.

Leaderboard Performance (Bonus Points)

A leaderboard on Gradescope will track the rankings based on Spearman correlation and recall scores. Bonus points will be awarded based on final leaderboard rankings:

- 1st place: +10% bonus points
- 2nd - 3th place: +5% bonus points
- 4th - 6th place: +3% bonus points

Note that the bonus points here are for the whole hackathon project.

Final Grade Breakdown

Category	Description	Weight
Spearman Correlation	Measures ranking consistency between predicted and actual mutation effects in the test set	30%
Top 10 Mutations	Measures the quality of the top 10 mutation recommendations provided by each participant's model	30%
Final Report	Each group’s final report describing their querying, training, and model architecture selection	40%
Leaderboard Bonus	Bonus points awarded to the top participants on gradescope’s leaderboard	Up to 10%

Supplementary Information

Deep Mutational Scans (DMS) data

Proteins are essential molecules in biological systems, and they are made up of chains of amino acids. Changing even a single amino acid in a protein—called a mutation—can affect its function, sometimes improving it and sometimes damaging it. Deep Mutational Scanning (DMS) is an experimental technique that systematically mutates a protein and measures how each mutation affects its function. These experiments generate large datasets where each mutation is assigned a fitness score that reflects its effect on the protein’s activity. dms-view (link) is a useful interactive tool for visualizing DMS data, which you can use to make sequence logo plots and positive differential selections. Figure 1 shows an example of a DMS dataset measuring the inferred additive G-P map parameters for the GB1 protein. [1]. In reality, DMS can be applied in studying multiple properties of a protein, such as thermostability, identifying stabilizing mutations that enhance protein robustness for industrial and therapeutic applications. Another crucial application is in binding affinity, where mutation scans inform on how proteins interact with ligands, substrates, or other macromolecules, aiding drug discovery and antibody engineering. In this hackathon, we will provide a collection of DMS datasets that measure the fitness effects of thousands of mutations for a protein of our choice. Note that the DMS data we will be providing only includes single-point mutations.

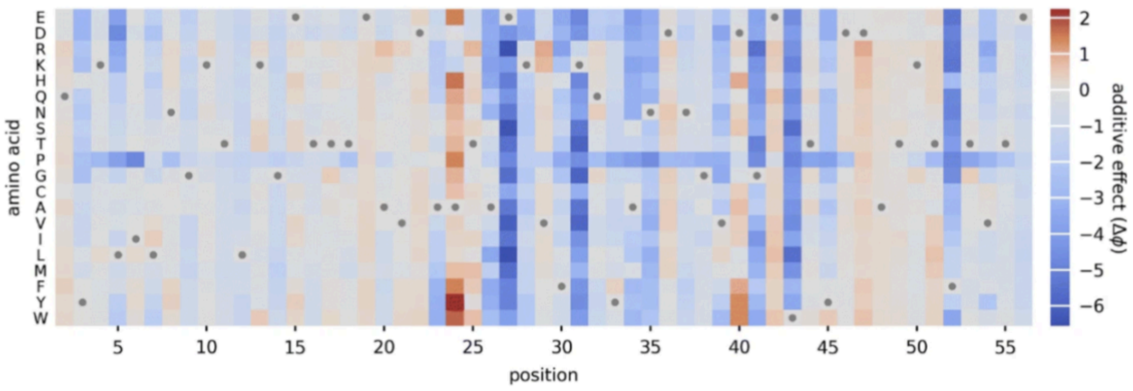


Figure 1: GB1 example DMS data measuring G-P map parameters

Protein Language Models (PLMs)

As you have learned earlier this semester, PLMs function similarly to natural language models like GPT, but instead of predicting words in a sentence, they predict and learn patterns in protein sequences. Given an input protein sequence, a PLM:

1. Encodes the sequence into a numerical representation (embedding).
2. Learns relationships between amino acids based on vast amounts of protein sequence data.
3. Uses its learned knowledge to infer properties such as mutation effects, structural stability, and function.

In this hackathon, you can choose from, but not limited to several of the state-of-the-art PLMs:

- ESM-1v – A mutation effect predictor trained on evolutionary sequences.
- ESM-2 – A general-purpose model that can be fine-tuned for mutation effect prediction.
- ProtT5 – A transformer-based model that generates contextual embeddings useful for training machine learning classifiers.

- Saprot – A large scale general purpose protein language model that introduces the concept of a “structure-aware vocabulary” that integrates residue tokens with structure tokens.

Here, we only provided some examples of the models you can use for this hackathon. You are welcome to select other models of your choice. One thing to keep in mind during
training is the size of the model. For example, ESM2 offers 3 billion and 15 billion models, which might take a decent amount to train. So we recommend using a smaller model or applying parameter-efficient tuning methods.

Training

You have the flexibility to choose their modeling approach, including:

1. Fine-tuning Protein Language Models (PLMs): Utilizing models like ESMs, ProtT5, or SaProt etc., which have shown improved performance in mutation effect prediction upon fine-tuning.

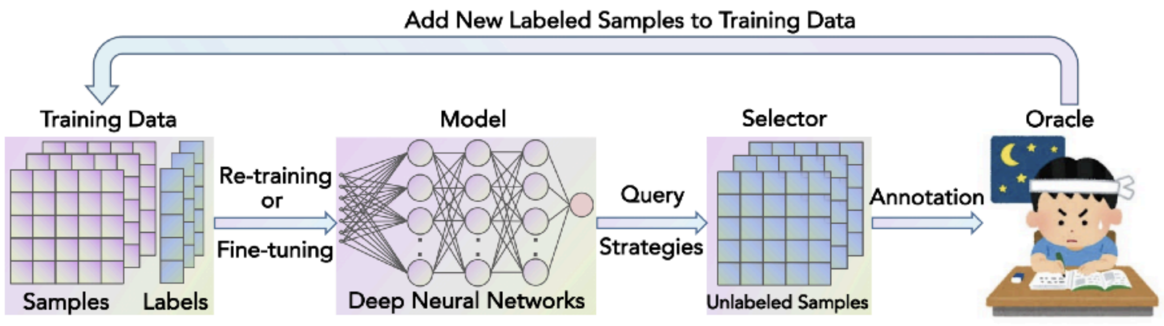
2. Parameter-Efficient Fine-Tuning (PEFT) [8]: Applying methods such as LoRA to adapt large PLMs efficiently, making them accessible even with limited computational resources. Using PEFT for PLM finetuning can also prevent overfitting, which in some cases can outperform finetuning the entire Model.

3. Traditional Machine Learning Models: Other than fine-tuning protein language models, you are also welcome to explore other traditional ML methods such as regression, you can encode the protein sequences using one-hot encoding, ESM embeddings, or other encoding methods, and train your own model. Some architectures you can consider include:
• CNN – Convolutional neural networks
• RNN – Recurrent neural networks
• GNN – Graph neural networks
• regression-based methods
• MLP - Multi-layer perceptrons

Active learning

Active learning is a machine learning paradigm where the model iteratively selects the most informative data points to label, aiming to improve performance with fewer labeled instances. In the context of protein mutation prediction, this approach is particularly valuable due to the vast sequence space and the experimental costs associated with obtaining fitness data. Understanding active learning strategies can be beneficial in this scenario. For instance, the study ”Active learning-assisted directed evolution” [9] demonstrates how active learning can efficiently navigate protein sequence space to identify beneficial mutations. This paper introduces the main difference between DE (direct evolution) and ALDE (Active learning assisted direct evolution). DE focuses on exploring a small, local region of a sequence while

ALDE alternates between collecting wet lab fitness data and training an ML model to prioritize new sequences to screen in the wet lab, which is directly related to this hackathon. We highly recommend that you read it carefully. The other survey papers are optional.



Above is a figure from a survey on active learning [10] that illustrates the overall pipeline of active learning in general. The pipeline starts with a model that is trained on an initial training dataset. Then query strategies are applied iteratively to select the most informative and representative samples. These data, along with their labels are then used for further retraining or fine-tuning of the active learning models. Active learning achieves competitive training performance with less annotation costs and computational resources. Another active learning survey paper [11] provides a more in-depth investigation in active learning.

A key question in this hackathon is "how to select the most informative data points to label" to maximize model performance, here are a few approaches that you can consider but you are also welcome to explore other approaches:

- **Random Selection:** A baseline approach where mutations are chosen at random without considering informativeness. This method provides an unbiased sample but is often inefficient in exploring sequence space.
- **Greedy Selection:** Prioritizes mutations expected to yield the highest immediate fitness improvement. This approach might risk getting stuck in local optima, especially in rugged, non-linear protein fitness landscapes with epistasis.
- **Upper confidence bound (UCB)**: Commonly used in Bayesian optimization, considers both the predicted fitness and the confidence of that prediction. For more details on this you can check out [15]
- **Thompson Sampling:** Another acquisition function where the next input to evaluate is obtained by drawing a sample (function) from the posterior distribution of the objective function and selecting the point that maximizes this sample.
- **Expected improvement (EI):** An acquisition from bayesian optimization that selects the next points that is expected to output the best observed outcome so far. It has a closed form solution based on the posterior mean and variance.
- **MLDE (Machine learning directed evolution):** The main logic of MLDE is to use in silico method to expedite screening and optimization process. [13] and [14] might be useful to check out.

- **Uncertainty based approaches:** Hie et al provides an approach to use uncertainty to prioritize combinatorial mutants to avGFP based on predicted fluorescence in their paper “Leveraging Uncertainty in Machine Learning Accelerates Biological Discovery and Design [12] This paper might provide you some insights on selecting data points by quantifying prediction uncertainty.

References

[1] Ammar Tareen, Mahdi Kooshkbaghi, Anna Posfai, William T Ireland, David M McCandlish, and Justin B Kinney. Mave-nn: learning genotype phenotype maps from multiplex assays of variant effect. Genome biology, 23(1):98, 2022.

[2] Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood Van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, et al. Proteingym: Large-scale benchmarks for protein fitness prediction and design. Advances in Neural Information Processing Systems, 36, 2024.

[3] Sam Gelman, Sarah A Fahlberg, Pete Heinzelman, Philip A Romero, and Anthony Gitter. Neural networks to learn protein sequence–function relationships from deep mutational scanning data. Proceedings of the National Academy of Sciences, 118(48):e2104878118, 2021.

[4] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. BioRxiv, 2022:500902, 2022.

[5] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. Advances in neural information processing systems, 34:29287–29303, 2021.

[6] Ahmed Elnaggar, Michael Heinzinger, and Christian Dallago. Prottrans: Toward understanding the language of life through self-supervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44:7112–7127, 2021.

[7] Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. bioRxiv, pages 2023–10, 2023.

[8] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. arXiv preprint arXiv:2403.14608, 2024.

[9] Jason Yang, Ravi G Lal, James C Bowden, Raul Astudillo, Mikhail A Hameedi, Sukhvinder Kaur, Matthew Hill, Yisong Yue, and Frances H Arnold. Active learning-assisted directed evolution. Nature Communications, 16(1):714, 2025.

[10] Dongyuan Li, Zhen Wang, Yankai Chen, Renhe Jiang, Weiping Ding, and Manabu Okumura. A survey on deep active learning: Recent advances and new frontiers. IEEE Transactions on Neural Networks and Learning Systems, 2024.

[11] Alaa Tharwat and Wolfram Schenck. A survey on active learning: State-of-the-art, practical challenges and research directions. Mathematics,11(4):820, 2023.

[12] Hie, Brian, Bryan D. Bryson, and Bonnie Berger. "Leveraging uncertainty in machine learning accelerates biological discovery and design." *Cell systems* 11.5 (2020): 461-477.

[13] Qiu, Yuchi, Jian Hu, and Guo-Wei Wei. "Cluster learning-assisted directed evolution." *Nature computational science* 1.12 (2021): 809-818.

[14] Tran, Thanh VT, and Truong Son Hy. "Protein design by directed evolution guided by large language models." *IEEE Transactions on Evolutionary Computation* (2024).

[15] Desautels, Thomas, Andreas Krause, and Joel W. Burdick. "Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization." *J. Mach. Learn. Res.* 15.1 (2014): 3873-3923.