Hackathon Preparation Exercise - 2025

Due - **19/06**
+5 bonus for submitting before the Hackathon (**17/06 00:00**)

1. Overview

This warm-up exercise is meant to help you prepare for the Hackathon. Most of the code is provided for you. The exercise walks you through the full analysis pipeline implemented in:

- '**ex4.py**' - sequence-level pipeline.
- **'structure_analysis.py'** - structure-level pipeline.

Your job is to fill in every **TODO** in the code, tune the hyperparameters, and answer reflection questions that demonstrate an understanding of what each processing step does, how the design choices affect performance, and how to interpret the plots you generate.

2. Files & Folders

- ex4.py - main script, loads peptides, gets ESM-2 embeddings, performs clustering/visualisation, distance baseline, trains a Dense NN classifier.

- structure_analysis.py - calculates peptide centre-of-mass (COM) distances and peptide pLDDT for AlphaFold2 models (all the models are aligned to a reference structure) and makes ROC/box plots.

- pep_utils.py - helper functions for loading CSVs, Euclidean distances, data split.

- esm_embeddings.py - wrapper around esm2.

- neural_net.py - a minimal fully-connected network + training/inference helpers (implemented with pytorch)

- clustering.py - k-means clustering and t-SNE dimensionality reduction.

- plot.py - all plotting utilities (heatmap, box/ROC/2D scatter).

3. Pipeline Walk-through

In this section we explain the workflow of the scripts. Most of it is implemented for you already and in section 4 we list exactly what you have to implement yourself. Please make sure that you

go over all the code and understand exactly what each function and section does - you will need it for the Hackathon. Also make sure that you understand the purpose of each plot and what information it provides for us.

ex4.py

1. ESM parameter block, you will tweak these parameters:

```
# TODO: play with these parameters
chosen_embedding_size = 1280  # ESM embedding dim (320-5120)
chosen_embedding_layer = 33  # which transformer layer to take
chosen_test_size = 0.25  # train/test split
```

2. Load peptide CSV, lists of positive & negative sequences.
3. Load the ESM2 model of the chosen embedding size.
4. Visual sanity check: plot per-residue heatmaps for the first positive & negative peptides.
5. Get sequence embeddings for all peptides.
6. Clustering & dimensionality-reduction: cluster the embeddings with the k-means algorithm, reduce the embeddings to 2D using t-SNE for visualization, and color by true labels and k-means clusters.
7. Train/test split.
8. Baseline classifier based on “distance” score, Positive scores -> more positive-like.
9. Plots for the baseline, ROC + boxplot.
10. Simple Dense-network classifier training, build and train a simple network and chose the following hyperparameters:

```
# TODO: Select parameters for the network
batch_size = 64
epochs = 50
lr = 1e-3
hidden_dim = 128
dropout = 0.2
```

11. Plots for the Neural network, ROC + boxplot.

structure_analysis.py

1. Load reference native complex (PDB 6cit) and compute its peptide COM.
2. For every AlphaFold model in “structures/af_positives” (label 1) and “structures/af_negatives” (label 0):

    a. Load structure.
    b. Get peptide atoms.
    c. Compute average pLDDT confidence of the peptide.
    d. Compute the distance between the peptide COM and native peptide COM.

3. Convert distances to score (-distances) so that higher -> better (for ROC).
4. Plot two ROC curves, COM-distance score and pLDDT score.
5. Plot two boxplots for the same metrics.

4. The *exact* things you should implement

file	**line**	**description**
`esm_embeddings.py`	62	Generate per-sequence representations via averaging (one liner)
`ex4.py`	33	in `simple_score`, complete the negative-score analogue of the positive line (one liner)
`structure_analysis.py`	87	compute and append distance (`dists.append(...)`)
`ex4.py`	41-43, 101-105	tune all “play-with” ESM hyper-parameters at the top of main & the training block.

5. What you need to submit

1. Completed code – all **TODOs** solved, runs end-to-end without errors.
2. **Eight** plots from ex4.py (2 embedding heatmaps, 2 2D scatter plots, ROC and boxplots for baseline & NN).
3. **Four** plots from structure_analysis.py (COM & pLDDT ROC + boxplots).
4. **Short answers** to the questions below (bullet points are fine).
5. **A signed commitment from all group members** that they have gone through the entire exercise and its entire solution.

Submit it directly to the GitHub repository you are using for the Hacathon.

6. Reflection Questions

Answer the following questions shortly **(1-2 sentences is enough)**.

<u>Embedding layer/size</u>

1. How did changing `chosen_embedding_layer` (e.g. 9/20/33) affect the baseline AUC?
2. Does a larger model always lead to improved performance?

<u>Distance baseline</u>

3. Explain in your own words why `score = log1p(dist_to_neg) − log1p(dist_to_pos)` distinguishes classes.

<u>Neural-network classifier</u>

4. Report the best test-set AUC you achieved and the hyperparameters that led to it.
5. We used the ESM sequence embedding for training a simple Dense classifier network.
    a. Can you think of alternative or additional inputs for the network that might improve its performance?
    b. Can you think of alternative architectures or classification algorithms? (not necessarily neural networks).

<u>Unsupervised structure</u>

6. Look at the t-SNE plot coloured by true labels. Are the two classes separable in 2D? Does k-means find meaningful clusters?

<u>AlphaFold COM analysis</u>

7. Which metric (COM distance vs mean pLDDT) is a better discriminator? Support with ROC AUC values.