Below is my detailed reasoning and methodology for ML task: Protein Structure Classification using the CATH database

My Problem Statement Understanding:  
Given protein pdb structures, sequences and information such as topology and superfamily homology, how can we accurately classify proteins into their respective architecture (in this case since there are 10 classes of architecture, it was a multiclass classification task).

Methodology:  
I worked on a few methodology and approaches to perform this classification task.

The main simplistic idea I carried with me initially is to split the dataset into train, test, and validation dataset first, in a way it avoids data leakage (as made aware of in the challenge).

Afterwards is the preprocessing step. In this step, I parsed the data using bio python’s inbuilt parser library. I initially tried parsing the data using biounit parser (a helper function for the model architecture I borrow from git), however due to time constraint, I moved with bios parser parsing.

I am reporting below the detailed steps I have taken to solve this task.

Under the ‘Drafting the Skeleton’ Segment, I thought of creating encoding tokens using the sequences and parsed pdb files to perform the classification task. I encoded the sequences using a Pre Trained ProtBert model and encoded the sequences.

Secondly, the worked to implement a pre trained GNN model to encode the graph structure.

Before doing that, I parsed the pdb files using bio python’s parser file. Then I implemented it to centralize the parsed data, after which I converted it into a graphical structure (representing node and edges to pass it into a GNN architecture and encode it.

However, since my computation operators of 2 core CPU space and GPU offered by Colab, the execution surpassed more than 8hrs and was still running and hence I terminated this methodology, as it was too slow (the tokenization process itself).

This led me to methodology two: “Attempt to Convert protein structures to images and train a CNN or GCN”. In this approach, my restriction in working on the colab file prevented me from being able to use the standard plymol3 library image conversion (as that specific function is not supported in the colab environment). And since other libraries too were time consuming to set up, I moved to a different methodological approach.

Before the next approach, I significantly worked in identifying the gaps in the structure file and sequence file.

In order to find gaps in the sequence file, I searched for homology sequences on the Uniprot database, mapping the pdb\_id of the respective proteins. I tried to fetch data from NCBI’s protein database but due to serve issues at the time, the fetching process for each of the pdb\_ids, including the wait time, surplussed more than 4 hrs and was still running, hence I decided to terminate the execution and the approach.

I also tried identifying using the Modeller library (again the environment of running it via colab was not supportive and hence failed to fetch the information). Using BioRepair too was also unsuccessful as it required a license key. My idea was to compare the sequences from what was present in the csv file to other homology sequences in a different sequence and add that information (hence repairing to the gap after identification).

Then I decided to pursue identifying gaps in the protein structure over sequences. For the protein structure, the main way I was able to capture the gap was via the residue jumps.

After that, I sought a model that inherently deals with these gaps (takes into account) and provides an output.

This led me to my third methodology to solve this task. Which is to load the sequence data, perform encoding (preprocessing) and then implement the ESM transformer, with a linear classification layer to get the sequence (adding other features such as the top;pgy and superfamily information).

Subsequently, I would perform preprocessing on the structured dataset (parse it, centralize it and convert it to a suitable structure and pass it to a GNN model and get the weighted classification). Finally I would take the weighted average of the outputs of both the ESM transformer and ProteinMPNN and get the solved classification. SInce I planned to use these pretrained models, I wanted to implement the Lora hyperparemeter tuning to fine tune the results I would get.

My other approach was to apply a diffusion model to repair the protein structure or use OpenFold AI, however due to lack of compute power and time, I did not explore the area completely.

Since I first started off to implement the PorteinMPNN model first and later the ESM model for the sequence data, upon studying the model structure, I found that this model is able to take both sequence and graph structure data as input.

Hence, I decided to go about this multimodal model. I didn't write this model architecture from scratch but adopted it from github (as provided in my colab notebook).

I chose this model as it combined both sequence and structure, like a message-passing neural network approach. It integrates sequence with structure data through the use of specialized featurization layers and a combination of encoding and decoding layers.

The model first inputs protein structure (coordinates) and sequence into a graph representation where nodes represent amino acids and edges represent spatial and sequential relationships. Hence, I had to parse the PDB files accordingly. Depending on the ca\_only flag, either all backbone atoms (N, CA, C, O) are considered or only CA atoms for node features. Edge features were computed based on the distance and orientation between residues, enriched with radial basis function (RBF) kernels to capture different distance scales. ANd I accounted for centralization here as well.

Initial embeddings were created for nodes (amino acids) and edges (relations between amino acids) using linear layers. This initial featurization translated raw data into a processable form suitable for deep learning models in general.

The encoder layers process the graph, updating node representations by aggregating information from their neighbors (other amino acids), which captures the context of each residue within the protein structure.

In the decoding layer, the model iteratively predicts the sequence of the protein, using masked attention to ensure that the prediction for each position can only use information from preceding residues, mimicking the autoregressive property of sequences.

My main challenges were adapting the preprocessed data to the model’s input architecture, which is mostly covered in my second submitted notebook. As the preprocessing steps demanded by this model were heavy, for each set it took about 20-45 minutes in average to compute the preprocessing for each.

To make the process faster, I implemented parallel processing of the loading and parsing of the data, and tried to utilize the available colab gpu on cpu (which required me to implement ‘spawn’ methods on the colab, which later on heavily interfered later, when I tried to laid my tensors onto the device (on gpu). Hence, I had to completely disregard and finally run the entire process on CPU (as calling spawn multiprocessing once, even if removed interferes until and unless the kernel is restarted or reset).

Hence unfortunately, I ran out of time to complete my model training and esure the tensor dimensions of my data inputs doesn’t interfere with the model inputs. I unfortunately didn’t have the authority to overwrite the model architecture without completely adding it to my system path.

Conclusively, I believe the ProteinMPNN model which I heavily attempted to implement is promising for this task, as it takes care of the structure gaps, takes in both sequence and graph structure and finally provides a classification output at the end.

I would really like to explore and satisfactorily implement or tweak the architecture to include edge features and perform LORA hypermater tuning.

Identification of haps and yo perform homology modeling succinctly would be other areas where I would really like to dedicate more time to the anticipated results.