# ML Researcher Hands-On Challenge

Jibran Bohra

December 9, 2023

**Abstract**

This report provides a concise overview of the code development process, emphasizing the crucial functions and scripts involved. It serves as a comprehensive guide to the methodology adopted, facilitating a clear understanding of the implemented features and their respective outcomes.

## Contents

# 1 Getting Started...

The provided `iPython` notebook served as the foundation for an Object-Oriented code development process. The resulting script, named `main.py`, is designed to preprocess an arbitrary dataset for utilization in a machine learning model. The key functions implemented in `main.py` are as follows:

- `generate_features()`: This function reads structural data, including coordinates, occupancy, and $b$-factor, from `.pdb` files. The extracted information is then stored in `all_data.json`.
- `preprocess()`: This function gathers columns from `all_data.json` for preprocessing. Specifically, it tokenizes protein sequence data, normalizes protein structure data (e.g., coordinates), and normalizes numerical data (e.g., coordinate mean and standard deviations, occupancy, $b$-factor). Additionally, class label data is prepared for one-hot encoding. The preprocessed data is saved in `structure.npy` and `preprocessed.json` for subsequent use.

# 2 Methodology

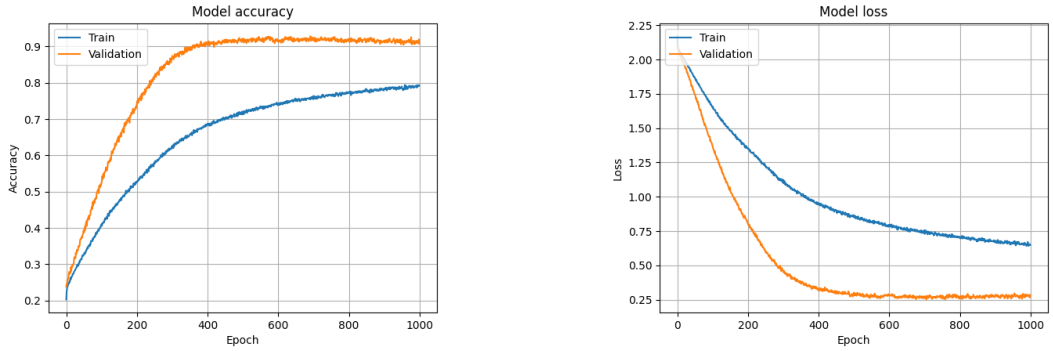
The development process is thoroughly documented within the `development-process/` folder, housing scripts that illuminate significant milestones, including:

- `ann-labels`: An Artificial Neural Network (ANN) primarily employed as a sanity check. This ANN takes *class* and *architecture* as inputs, aiming to classify protein architecture. The model achieved 100% accuracy, on par with expectations.
- `ann-numerical`: An ANN dedicated to classifying protein architecture based on numerical data (as mentioned above). However, both training and validation accuracy were limited to approximately $\approx 25\%$.
- `cnn-structure`: A Convolutional Neural Network (CNN) designed to derive correlative features from `structure.npy`, encompassing $x$-, $y$-, and $z$-coordinate data. Despite training accuracy generally being higher, validation accuracy plateaued at $\approx 20\%$, indicating a tendency to overfit, unlike `ann-numerical`.
- `lstm-sequence`: A Long Short-Term Memory (LSTM) model utilized for processing tokenized protein sequence data. Model exhibited a capped validation accuracy of $\approx 20\%$ and a predisposition towards overfitting, much like `cnn-structure`.

# 3 General Model and Results

Leveraging the advancements outlined earlier, a comprehensive model was constructed to classify protein architectures using protein sequence, protein structure, and various numerical data. The architecture of this model is highlighted in `all_data _classifier.py`.

To enhance the dataset, data points were augmented by a factor of eight. Notably, structural and numerical data underwent systematic rescaling, while label and sequence data were straightforwardly replicated. The incorporation of data augmentation, along with the inclusion of dropout layers and regularization, aimed to alleviate tendencies toward overfitting. The training and validation performance of the general model over 1000 epochs is illustrated in Figure 1.




(a) (b)

Figure 1: Model performance encapsulated by Figures 1a and 1b demonstrates accuracy and loss, respectively, across 1000 epochs

Note that the relatively modest training performance can be attributed to the strategic use of dropout layers, which introduce random weight adjustments, coupled with the application of regularization techniques. Additionally, the validation accuracy levels off at around $\approx 92\%$ during the $400^{\text{th}}$ epoch, while the training accuracy continues to rise. This indicates that the peak model performance has been achieved and further training is likely to cause overfitting.

# 4 Discussion and Conclusion

In conclusion, the general model crafted for the classification of protein architecture achieves an accuracy of $\approx 90\%$ for unseen samples. To enhance performance and facilitate further development, it is advisable to employ a larger dataset. Additionally, it's crucial to acknowledge that the scope of this exercise was constrained by

hardware limitations. I recommend training future models with a dedicated GPU to unlock greater computational capabilities and potentially improve outcomes.

# References

[1] Ronak Vijay, *Protein Sequence Classification: A case study on the Pfam dataset to classify protein families.*