PBL Project: Prediction of enzymatic activity

Neo Mehmedinovic, Luis Telaak, David H. Nar, Daniel Poock

Abstract

Motivation: The rapid expansion of omics data has far outpaced the capacity for experimental protein annotation, making accurate computational prediction of Enzyme Commission (EC) numbers essential for understanding cellular metabolism. While traditional homology-based tools like BLASTp [5] remain competitive, their reliability declines significantly for enzymes sharing low sequence similarity with known databases. This project addresses the need for a computationally efficient machine learning framework that can accurately classify enzymes under strict sequence similarity constraints.

Methodology: We present a compact hierarchical pipeline consisting of models that perform three distinct tasks: binary enzyme detection (Task1Model), multi-class main EC class prediction (Task2Model), and their joint integration (Task3Model). Prioritizing efficiency, we utilize dimensionality-reduced ProtT5 [8] embeddings (128 dimensions) as information-rich, low-dimensional feature vectors. These enable our proposed feed-forward neural networks (FNNs) to be lightweight models to allow for quick and simple inference. A clustering pipeline was employed using a 30% sequence similarity threshold to enforce strict sequence similarity constraints and prevent data leakage between sets.

Results: Our proposed Task1Model achieved a Matthews correlation coefficient (MCC) of 0.82 for enzyme detection and our Task2Model a macro-F1 score of 0.74 for main class classification. Evaluated on the independent Price-149 benchmark, the framework maintained a weighted F1-score of 0.72, demonstrating stability across internal and external datasets. Ultimately, our project challenges the assumption that complex architectures and high parameter counts are essential for effective enzyme classification. We show that straightforward, lightweight models can also yield robust and competitive results.

1 Introduction

Enzymes play a key role in biology because their catalytic function enables reactions to proceed faster and more energy efficiently. Consequently, the accurate and systematical classification of enzymes has long been a primary objective, as it enables a deeper understanding of enzyme function and cellular metabolism [1]. The Enzyme Commission (EC) [2] numbers provide the standard solution for this challenge. They offer a four-level hierarchical nomenclature system that classifies enzymes according to the reactions they catalyze. Accordingly, a complete EC annotation entails predicting the specific fourth-level digit. Over the past two decades, the rapid growth of omics data [3], particularly proteomics, has far outpaced experimental amino acid sequence annotation. As a result, the accurate and efficient prediction of EC numbers has become essential for significantly improving our understanding of enzymatic functions and is now a central discipline in bioinformatics [2].

Several approaches exist to address the task. Early methods relied primarily on sequence alignment and homology, which proved effective. Tools such as BLASTp [5], introduced in 1990, remain competitive with state-of-the-art models in many settings. More recent approaches include Machine Learning, such as neural networks, to predict EC numbers directly from protein sequences. ECPred [6] formulates the assignment as a hierarchical prediction problem. Rather than treating the labels as single level classification, it divides the task into consecutive decisions at each EC level. As for neural-network-based approaches, DeepEC [2] combines convolutional neural networks (CNNs) with a homology-based step to better identify cases in which the classifier is uncertain. More recently, CLEAN [7] (contrastive learning-enabled enzyme annotation) learns a representation space using contrastive learning, which mitigates the issues data scarcity introduces. EC numbers are then assigned based on their distance in this embedding space. CLEAN reports better performance than commonly used baselines such as BLASTp.

These state-of-the-art models are increasingly used to provide reliable computational enzyme annotation, but the practical use they provide depends on their ability to generalize the training data beyond redundancy and handle any class imbalance. In our project *Prediction of enzymatic activity,* we particularly focused on tackling these challenges while prioritizing computational efficiency. To this end, we developed three different predictive models, each tackling a distinct classification task:

- Task 1 (Binary classification): For a given protein, predict whether it's an enzyme or not
- Task 2 (Multi-class classification): For a given enzyme, predict to which main EC class (1-7) it belongs
- Task 3: Predict EC class for proteins predicted to be enzymes

Our proposed solutions leverage dimensionality reduction of ProtT5 [8] embedding vectors and strategic under-sampling to enable us to create an exceptionally compact model architecture. The resulting lightweight design significantly mitigates computational overhead and allows for rapid training and inference.

2 Material and Methods

For writing our models and the surrounding functionality, we used the Python programming language, primarily utilizing the libraries PyTorch [9], PyMMSeqs [10] / MMseqs2 [11], scikit-learn [12], pandas [13], and Biopython [14].

2.1 Metrics

To enable structured comparison and comprehensive assessment for given models, we report performance using established evaluation metrics. Due to our study comprising a binary (Task 1) and a multi-class classification (Tasks 2 and 3), we used task-specific scoring metrics.

2.1.1 Task 1 Metrics

We evaluate the binary prediction task using the Matthews correlation coefficient (MCC) as the primary metric. It incorporates all four confusion-matrix outcomes (TP, TN, FP, FN) and attains high scores only if both error types are low. The measure returns a value between -1 and 1, with -1 representing a perfectly incorrect classifier, 0 representing a random classifier and 1 representing a perfect classifier. This makes it a robust and balanced metric for binary classification [15].

2.1.2 Task 2 and 3 Metrics

For the multi-class predictions, we evaluate model performance using the macro-F1. It is computed by taking the arithmetic mean of all class-wise F1 scores. Because every class contributes equally, the macro-F1 is well-

suited for strongly imbalanced datasets, especially in comparison with the MCC for multi-class classification. It produces values in the 0 to 1 range. High scores only occur when both precision and recall are consistently high across all classes. Furthermore, it offers some chance-correction and is unaffected by false positive and false negative distribution within a given class [16].
The macro-recall is implicitly tracked in the row-normalized confusion matrices by averaging the diagonal percentage values but was not explicitly tracked in the evaluation.

2.2 Task 1

2.2.1 Dataset Selection

We retrieved the initial dataset from the UniProtKB database [17], filtering for proteins with experimentally proven functions using the query *cc_function_exp:**. This guaranteed high-quality labels, enabling the model to train on data with biological evidence rather than prior computational annotations. The resulting raw dataset consisted of 104,729 protein sequences, showing that there was still sufficient data for model training despite primarily focusing on data quality for dataset selection.

2.2.2 Dataset Preprocessing

To prepare the dataset for training and evaluation we employed the following three preprocessing steps to ensure robustness and mitigate bias:

1. Redundancy Reduction: To prevent data leakage between train and test sets, we reduced sequence redundancy using the clustering tool MMseqs2 [11]. We applied a sequence similarity threshold of 30%, while clustering enzymes and non-enzymes separately. This ensures that no sequences with different labels get clustered together and that the model learns to generalize across diverse protein families rather than memorizing similar sequences.
2. Outlier Removal: Protein sequences show significant length variations. Extremely short sequences or sequence fragments may lack sufficient information for classification, while extremely long sequences increase computational cost and may introduce noise. To address this issue, we removed sequence length outliers by excluding the shortest and longest 2% of the distribution (sequences shorter than 17 and longer than 1890 amino acids were removed).
3. Final Dataset Split: The preprocessing resulted in a dataset of 39,335 unique proteins. We split the dataset into a training set (85%, 33,434 sequences) and a test set (15%, 5,901 sequences). The test set was not shown to the model during the hyperparameter tuning and training phases to serve for an unbiased evaluation.

2.2.3 Feature Extraction

To transform the sequences into machine readable input, we used embeddings from the pre-trained protein language model ProtT5 (XL-UniRef50) [18]. While we initially tested classical encoding methods such as One-Hot-Encoding or Amino Acid Composition, our experiments showed that these approaches failed to match the predictive performance of pLM embeddings. Raw ProtT5 embeddings are high-dimensional vectors with 1024 features per protein. Using Principal Component Analysis from the scikit-learn Python library [12], we reduced the dimensions from 1024 to 128 to preserve the most relevant information while making it possible for us to build a more compact model which helps to prevent overfitting and directly tackles one of our main goals with our task.

2.2.4 Hyperparameter Tuning

We optimized our model configuration using a manual, iterative search process utilizing a stratified 5-fold-cross-validation. The stratified approach makes sure that each fold has the same class distribution to prevent sampling bias. During this optimization phase, we tested several hyperparameters to find a combination for which the model doesn't overfit without compromising performance. These can be split into two main categories:

1. Model Architecture: We experimented with different hidden layer dimensions (e.g. 32, 64, 128 neurons) and depths in combination with different dropout values, normalization layers, and different activation functions.
2. Training: We tuned the training dynamics by adjusting learning rate, batch size, number of epochs. Additionally, we tried out different loss functions and optimizers. Lastly, we experimented with the weight decay parameter (L2 regularization) as a constraint against overfitting.

2.2.5 Task1Model architecture

Based on our cross-validation analysis we finalized our model as a fully connected feed-forward neural network (FNN) with 128 input neurons for the dimensionality-reduced feature vectors. The architecture depicted in Figure 1 shows the funnel design. The first hidden layer has 128 neurons with a dropout of 0.5 and the second hidden layer has 64 neurons with a dropout of 0.3 (indicated by the red crosses).
We trained the model using the Adam optimizer with a learning rate of 0.001 and based on the Binary-Cross-Entropy loss function. For the final training run we used early stopping to receive the model at the time of the best performing epoch. Our final Task1Model was retrieved from the checkpoint recorded at epoch 47. The final architecture can be seen in Figure 1.

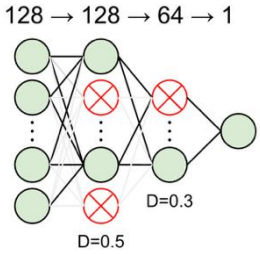

Figure 1: Task1Model architecture

2.3 Task 2

Task 2 involves developing a model capable of multi-class classification, assigning the first level EC number to given enzyme sequences.

2.3.1 Dataset Selection

We use the Swiss-Prot [17] subset of experimentally validated enzymes comprising 41,406 sequences. The raw label distribution is strongly skewed towards EC classes 1-3

2.3.2 Dataset Preprocessing

To control homology-driven leakage and class imbalance, we applied a fixed preprocessing pipeline, resulting in a final dataset of 7,850 sequences. Its distribution is additionally shown in Figure 2. This workflow strongly resembles the methodology employed in Task 1 and involves the following steps:

1. Multi-label sequence removal: First, we removed every enzyme sequence annotated with multiple first-level EC numbers to prevent label ambiguity.
2. Redundancy reduction: To reduce data leakage and redundancy as well as avoid unrealistic test performance due to homologous sequence overlap between train and test set, we grouped sequences by EC number and clustered them separately at 30% sequence identity utilizing MMseqs2 [11].
3. Outlier removal: Sequences shorter than 49 amino acids and longer than 1910 amino acids were discarded. This removed fragments and extreme multi-domain/outlier proteins that can dominate embedding variance and destabilize training.
4. Under-sampling: Since class distribution remained strongly imbalanced, we applied under-sampling with a hard cap of 2000 sequences per EC split, resulting in EC classes 1 to 3 being reduced to 2000 sequences each while retaining all available data for the minority classes. This cap was selected based on various cross-validation scores.
5. Final Dataset Split: The preprocessing resulted in a dataset of 7,850 enzymes. We split the dataset into a training set (85%, 6,672 sequences) and a test set (15%, 1,178 sequences).

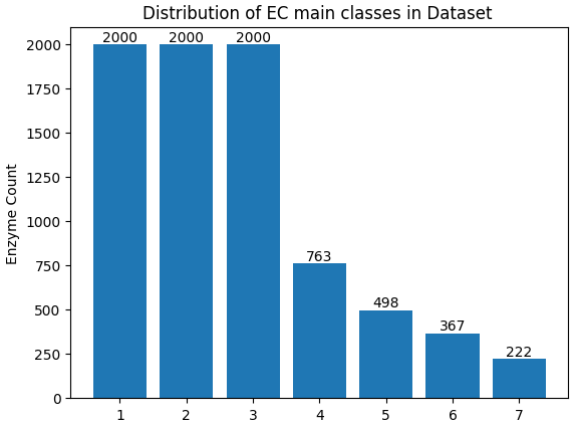

Figure 2: Task 2 dataset class distribution

2.3.3 Feature Extraction

Following preprocessing and splitting, the dataset underwent a feature extraction pipeline nearly identical to that of Task 1. The only difference is that we apply an additional homology filter to the train set prior to embedding. Specifically, sequences sharing over 30% sequence identity with the Price-149 [19] dataset were removed from the train set. This step allows us to adequately compare our model performance on the Price-149 dataset to state-of-the-art models as we do in 4.3. Our Feature Extraction pipeline is additionally shown in Figure 3.

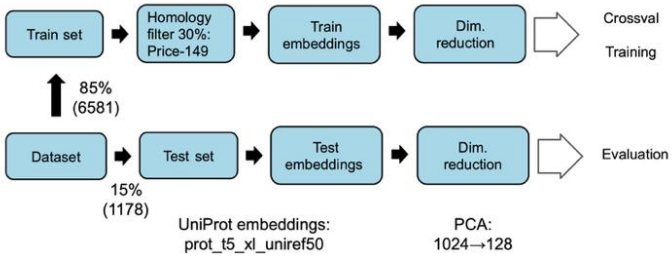

Figure 3: Task 2 feature extraction pipeline

2.3.4 Hyperparameter Tuning

To maximize the potential of our model, we tuned hyperparameters over cross-validation (5 folds) runs to increase macro-F1 while limiting overfitting, quantified by the train to validation generalization gap. This proceeded via the following steps:

- Optimization setup: Hyperparameter search was performed using Optuna [20] in combination with the tracking tool Weights & Biases [21], which enabled high-quality insights into each cross-validation run.
- Objective: We passed Optuna the objective to maximize macro-F1, while monitoring and limiting the generalization gap to avoid configurations that overfit too heavily.
- Search space: Our broad search analyzed over 10 different hyperparameters, including hidden layer size, dropout rates, learning rate, weight decay, and the under-sampling cap, which was evaluated between caps of 500 to 3000 sequences per class.
- Outcome: Using the optimal configuration of hyperparameters, the model achieved an average macro-F1 of 0.75 over all cross-validation splits.

2.3.5 Task2Model architecture

The final Task2Model architecture was derived from the optimal configuration from our hyperparameter optimization. Like our Task1Model, it is a feed-forward neural network (FNN) trained on our PCA reduced 128-dimensional ProtT5 features, which serve as input. The network uses a single hidden layer with 256 units and a linear output layer with 7 logits, corresponding to the 7 first-level EC classes. We applied a dropout of 0.12 and used Leaky ReLU for the activation function. In terms of training, we utilized cross-entropy loss and the AdamW optimizer with a learning rate of 0.01. Additionally, we employed a weighted random sampler to tackle train set imbalance. The final Task2Model architecture is visualized in Figure 4.

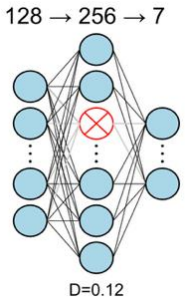

Figure 4: Task2Model architecture

2.4 Task 3

Upon establishing the optimal architectures and hyperparameters for the binary (Task 1) and multi-class (Task 2) models, our final objective was to integrate these components into a unified predictive framework. The aim was to construct a model capable of distinguishing non-enzymatic sequences (Class 0) from the seven EC main classes (Classes 1–7), effectively performing an 8-class prediction.

2.4.1 Dataset Selection

Since the Task 3 pipeline utilizes pre-trained components, no new training dataset was required. However, the construction of a valid test set was critical to prevent data leakage. Since the integrated pipeline relies on two models trained on different subsets of data, a simple concatenation of the Task 1 and Task 2 test sets would yield biased results. Specifically, sequences from the Task 2 training set might appear in the Task 1 test set, leading to an overestimation of the model's generalization capabilities.
To ensure a robust and independent evaluation, a specialized Task 3 test set was curated using the following protocol:

- Non-Enzyme Sequences: These were sourced exclusively from the Task 1 test set. Since the Task 2 datasets consist entirely of enzymes, this ensured there was no overlap or redundancy between the non-enzyme sequences and the data our Task2Model was trained on.
- Enzyme Sequences: These were selected from the Task 2 test set. To mitigate data leakage, we filtered these against the Task 1 train set, excluding any sequence sharing over 30% identity with it. We specifically chose this threshold to stay consistent with the homology reduction methodology applied in Tasks 1 and 2.

This led our final Task 3 test set to consist of 4191 sequences.

2.4.2 Task3Model architecture

Rather than training a Feed-forward Neural Network (FNN) with an eight-dimensional output layer, an architectural decision was made to implement a hierarchical pipeline (refer to Figure 5). This modular approach leverages the specialized performance of the previously developed models and integrates them into a sequential decision-making process for each amino acid sequence to be predicted:

1. Primary Classifier (Task1Model): An input amino acid sequence is first processed to determine its enzymatic status. If the model predicts "No Enzyme", the sequence is definitively categorized as Class 0.
2. Secondary Classifier (Task2Model): If the sequence is identified as an enzyme (Class 1) by the primary estimator, it is passed to our Task2Model, which assigns the specific main EC class (1–7).

This hierarchical architecture mirrors the logic of state-of-the-art tools such as ECPred [6] and DeepEC [2], both of which utilize discrete components for binary discrimination and specific functional assignment. However, the decision to implement this modular pipeline was largely pragmatic, aiming to maximize computational efficiency. By concatenating the pre-trained models from Task 1 and Task 2, we eliminated the substantial computational overhead and parameter tuning required to train a singular 8-class network from scratch, thereby preserving the validity of our prior optimizations.

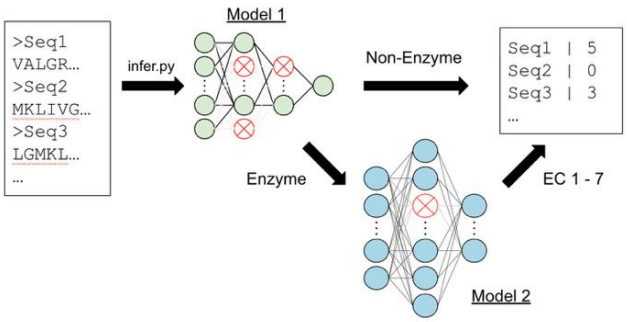

Figure 5: Task3Model pipeline architecture

2.5 Testing methodology

For a robust and realistic assessment of model performance, we utilized bootstrapping on our test set. This procedure helps mitigate sampling bias within a single test set split and allows us to calculate confidence intervals for our performance metrics. We performed 1000 bootstrapping iterations, resampling from the test set with replacement to generate sets of identical size to the original.

3 Results

3.1 Baseline Classifiers

To benchmark the performance of our Task 1 and Task 2 models, we selected several simple baseline estimators trained on the respective train set for comparison. We used the DummyClassifier as our random baseline as well as RandomForestClassifier, LogisticRegression and LinearSVC from the scikit-learn Python library [12] as basic classifiers.
These estimators provide a comparison base as they are easy to implement but also achieve relatively high scores compared to a random predictor. Logically, if our model performs worse than these baselines, the additional computational cost of our model is unjustified.
Additionally, these estimators cover a broad methodological range. This ensures that we compare our model to estimators that capitalize on fundamentally different architectural decisions and we don't overlook instances where the simple estimator excels due to its mechanics.
Therefore, by comparing our model to and outperforming these classifiers, we establish our model to have genuine predictive value over simple-to-implement estimators.

3.2 Model Performance Evaluation / Comparison

3.2.1 Task1Model Performance

For the final evaluation of our Task1Model we examined its performance on the Task 1 test set, which contains 5901 protein sequences. We begin by focusing on the row-normalized confusion matrix (refer to Figure 6). The diagonal represents the per-class recall values (e.g. the top left corner corresponds to the actual non-enzymes predicted as non-enzymes).

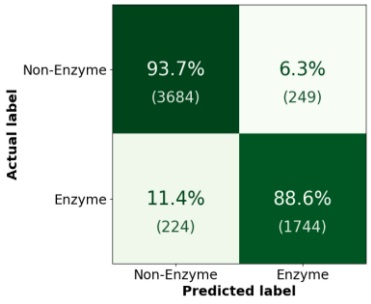

Figure 6: Task1Model confusion matrix

Consequently, our classifier correctly identifies 93.7% of non-enzymes and 88.6% of enzymes, demonstrating consistent high detection rates for both categories, not showing a significant bias towards one class. Furthermore, misclassifications are not evenly distributed. More specifically, 6.3% of non-enzymes are incorrectly predicted as enzymes (false positives), while 11.4% of enzymes are assigned to the non-enzyme label (false negatives). This indicates a tendency to avoid predicting the Enzyme label, unless sufficient evidence is present.
For a direct comparison to the baseline estimators introduced in 3.1, the test set performance was computed using the bootstrapped MCC (refer to Figure 7). Our Task1Model, shown on the far right, achieves the highest score, reaching an MCC of 0.82 with a standard deviation of 0.008 over the bootstrapping iterations. With this performance, it exceeds the classical approaches, including LinearSVC (0.74 ± 0.009), Logistic Regression (0.74 ± 0.009), and Random Forest (0.70 ± 0.010). As was to be expected, the random predictor scores an MCC of 0.
In summary, the consistently high class-wise detection rates, together with the respective MCC suggest that our Task1Model yields the most dependable and robust predictions on previously unseen sequences out of all simple classifiers. It achieves this even with a highly compact model architecture and computationally efficient training and inference pipelines.

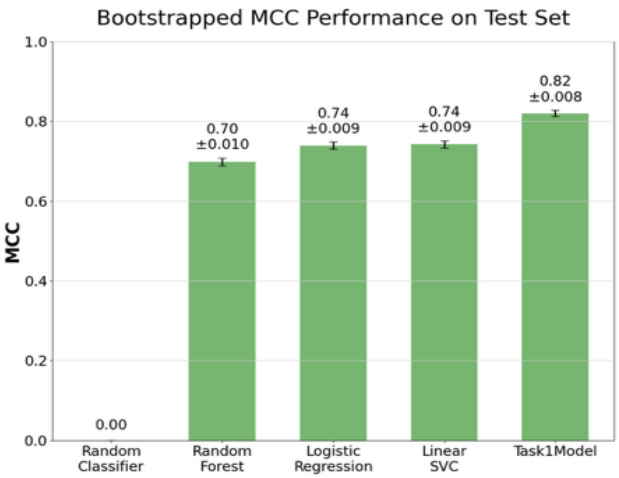

Figure 7: Task1Model performance

3.2.2 Task2Model Performance

To evaluate the inference capabilities of our Task2Model, we analyzed predictions from the Task 2 test set consisting of 1178 enzyme sequences. First, we examine the row-normalized confusion matrix (refer to Figure 8).

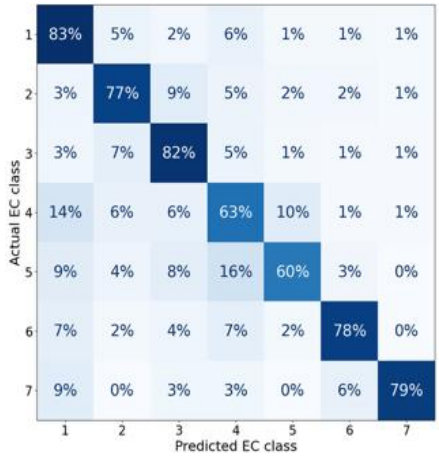

Figure 8: Task2Model confusion matrix

Most notably, the model proves to be efficient in correctly identifying the minority classes 6 and 7. With per-class recall values of 78% and 79% it almost reaches the same performance as for the majority classes 1, 2, and 3, with recall scores of 83%, 77%, and 82%, respectively. This establishes

our efforts in tackling classification bias resulting from training set imbalances to be effective. However, the model visibly struggles in classifying the remaining classes 4 and 5. It not only exhibits lower recall values of 63% (Class 4) and 60% (Class 5) but also reveals cross-class confusion. Specifically, 10% of Class 4 enzymes were mislabeled as Class 5, while 16% of Class 5 enzymes were incorrectly predicted as Class 4. Additionally, the model exposes a recurring misclassification pattern. Enzymes belonging to classes 4 to 7 are incorrectly assigned to Class 1, discernable by a distinct vertical band in the bottom left corner of the confusion matrix. Shifting the focus to the performance comparison, our model achieved a macro-F1 of approximately 0.74, which exhibits a substantial step-up in performance to the simpler estimators. The strongest competitor, the LinearSVC classifier, reaches a score of 0.6, while Logistic Regression (0.58) and Random Forest (0.44) yielded even lower scores. Our random baseline, the DummyClassifier, reached a macro-F1 of 0.06. These results highlight the complexity of this specific annotation task, implying the necessity for more complex Deep Learning architectures.

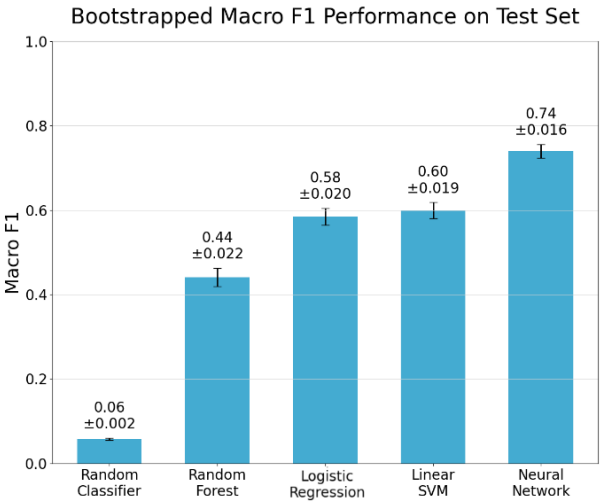

Figure 9: Task2Model performance

3.2.3 Task3Model Performance

Since the Task3Model isn't one single model like our previous FNNs, but rather a hierarchical combination of the two, it does not use a single, consistent training set. This makes it difficult to evaluate its performance directly against the baseline classifiers we used for our Task 1 and Task 2 comparisons. Effectively, we can't replicate training conditions for all models, possibly making a comparison statistically and methodologically inconsistent. For this reason, we focused our performance analysis on the row-normalized confusion matrix using the test set constructed in 2.4.2 which comprises 4191 sequences. The result aligns with our expectations as it closely represents a combination of the patterns observed in our Task 1 and Task 2 confusion matrices. It correctly predicts 93.7% of non-enzymes and just like our findings in Task 2, it encounters difficulty classifying EC classes 4 and 5.

Still, the confusion matrix does reveal key differences and provides further insights into the predictive performance particularly of our Task1Model. Specifically, EC classes 2, 3, 4, 5, and 7 are frequently misclassified as non-enzymes. These percentages may appear significant due to row-normalization, but this just represents the false negative instances of our Task1Model. However, the misclassification rate of EC Class 7 is the highest out of all classes, with 37.5% of these enzymes being incorrectly predicted as non-enzymes. This indicates a flaw in our Task1Model regarding Class 7 enzymes. Consequently, our Task3Model predicts Class 0 more often than the correct label for these sequences. Even so, this error may lack statistical significance as it could stem from having too little Class 7 representatives in our Task 3 test set. Additionally, the possibility exists that non-enzymatic proteins and EC Class 7 enzymes share inherent biological similarities.

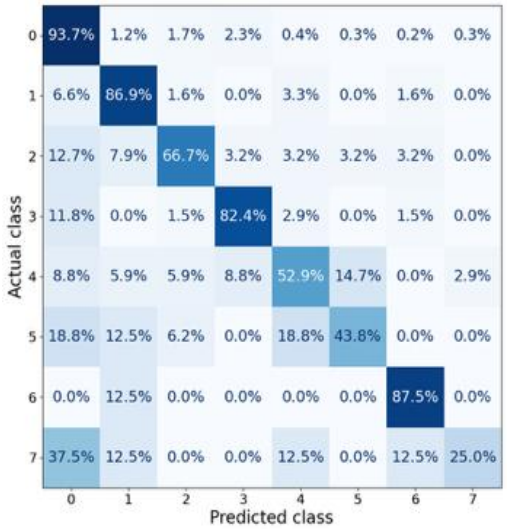

Figure 10: Task3Model confusion matrix

4 Discussion

4.1 Challenges

A significant portion of our efforts went into tackling class imbalance, trying to make our model as robust and unbiased as possible. Our goal was to avoid models that predict certain classes significantly better than others, as a model that favors specific labels for classification fails to address the real-world problem we're trying to tackle. In a practical setting, a model should be able to classify Transferases (EC 2) and Translocases (EC 7) equally as well. While the Task 1 training set did exhibit some imbalance, it becomes negligible when compared to the heavily skewed class distribution of the Task 2 training set, which was shown in 2.3.2. Among the variety of different methods tested, including weighted loss functions, weighted sampling techniques, under-sampling turned out to be the most effective in directly mitigating the effects of class imbalance.

A critical component of this strategy was the choice of an under-sampling cap, which required navigating the trade-off between dataset size and dataset balance. While a low under-sampling cap produces a more stable distribution, the training set would still lose valuable information. Conversely, choosing a high cap retains that information while introducing more imbalance. Ultimately, a cap of 2000 was identified as the optimal balance between these two factors. This choice is supported by the unbiased performance of the Task2Model on the test set shown in 3.2.2.

An additional challenge involved our hyperparameter tuning using Optuna [20]. Optimizing exclusively for raw performance frequently yielded model configurations that appear to heavily overfit the data. It was difficult to decide how much to prioritize performance in comparison to overfitting when choosing our final model. In the end, we still primarily based our model selection on raw performance, but we additionally implemented a constraint to ensure the generalization gap remains within acceptable limits. This choice aligns with the findings of Belkin et al. (2019) [22], which challenge the traditional view that sees a generalization gap as an immediate disqualifier of a model. Their research shows that in large deep-learning models, prioritizing raw test performance is often more important than minimizing the gap.

4.2 Limitations

Despite achieving reliable performance on our test sets, both Task 1 and Task 2 models remain limited in their capabilities, especially when compared to existing state-of-the-art methods. The Task2Model, in particular, works within a limited scope that could be expanded on in the future.

First, our model can only predict the main EC Class for a given enzyme. Established methods like CLEAN [7] on the other hand, are not only able to assign all four levels of the EC hierarchy but do so reliably.

Additionally, our model exclusively assigns one EC Class to an input sequence, as it is not designed to perform multi-label classification. While this is a limitation of our model, our project scope did not include this type

of classification. This was one of the reasons we explicitly removed all enzymes annotated with multiple EC classes from our training and test set. Lastly, our Task2Model performance appears to have hit a plateau within its current framework. Despite performing extensive hyperparameter tuning for over 5000 cross-validation runs, we didn't encounter a model with a better performance. This suggests that the limitations are structural rather than parametric. Specifically, the chosen architecture of an FNN seems to hit its performance ceiling. Therefore, exceeding our current performance would possibly entail a switch to a new model concept and design.

4.3 SOTA Comparison

To benchmark our Task2Model against state-of-the-art methods, we additionally tested our Task2Model on the Price-149 dataset [19], which we retrieved from the CLEAN GitHub repository [23]. It's a challenging dataset of 149 experimentally curated bacterial protein sequences used in prior studies like ProteInfer [24] and CLEAN [7]. It contains sequences where homology-based methods struggle, making it a useful dataset to evaluate model robustness [25].

We performed this evaluation in accordance with the EC-Bench framework [25] to ensure methodological consistency and meaningful comparison. As mentioned in section 2.2.3, we employed an additional homology filter based on the Price-149 dataset on our training data. We did this to ensure realistic and most critically comparable scores.

On this dataset, our Task2Model achieved weighted-F1- and macro-F1 scores of 0.72. Crucially, these scores underline the stability of our model. A macro-F1 of 0.72 on the Price-149 dataset remains closely aligned with our internal test set performance (macro-F1 of 0.74). This consistency indicates that our model is highly robust and successfully captured generalized functional features that remain valid even for difficult sequences.

When comparing weighted-F1-Scores with other state-of-the-art models on the first EC level, our model significantly outperforms DeepEC (0.35) [2] but struggles to keep up with newer models such as ECPred (0.87) [6] and CLEAN (0.95) [7]. Surprisingly, our model also gets surpassed by the homology-based tool BLASTp [5] which remains a formidable gold standard for top-level classification. Even at a 30% similarity threshold, BLASTp can often identify highly conserved local catalytic motifs that define the main EC classes. We are satisfied with these results as they confirm the achievement of a robust, generalized model capable of maintaining high performance under strict similarity constraints. It achieves these scores using an architecture that is substantially less complex and smaller than other state-of-the-art methods, effectively fulfilling our objective of computational efficiency and robust performance.

While there is room for improvement our model provides a reliable baseline for broad enzyme annotation.

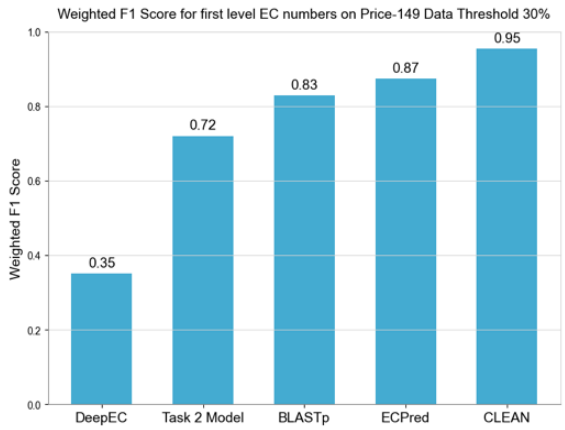

Figure 11: SOTA comparison on Price-149 dataset

5 Conclusion

This project demonstrates the validity and efficiency of a compact, hierarchical machine learning pipeline for the functional annotation of protein sequences. While current trends in bioinformatics often favor increasingly complex architectures, our results show that dimensionality-reduced ProtT5 [8] embeddings) combined with compact feed-forward neural networks (FNN) provide a robust and computationally lightweight solution. With a Matthews Correlation Coefficient (MCC) of 0.82 in enzyme detection (Task 1) and a macro-F1 score of 0.74 in main class prediction (Task 2), our model outperforms classical estimators such as Random Forest and Logistic Regression. The stability of these metrics on the independent Price-149 [19] benchmark (weighted/macro-F1 of 0.72) further validates the generalization capabilities of our architecture, even when subjected to a strict 30% sequence identity filter. This performance confirms that our model captures essential functional features beyond mere sequence homology, which is critical for annotating proteins where tools like BLASTp [5] reach their limits.

While limitations were observed in the classification of EC classes 4, 5, and 7, the overall consistency of our results indicates a strong baseline for further development. Future work should investigate whether integrating contrastive learning, as seen in CLEAN [7], or utilizing an entirely new architecture can further improve performance without sacrificing the efficiency of our lightweight design. Additionally, future enhancements should improve model resolution, particularly allowing the prediction of all four levels of the EC hierarchy, similar to state-of-the-art methods.

6 Contributions

Neo, Luis, David, and Daniel jointly conceived and designed the project. Daniel served as group lead, providing overall orientation and coordination. All authors contributed equally to model creation, analysis and interpretation of results, wrote the manuscript together, and approved the final version.

7 References

[1] Blanco A., Blanco G. 2017. Chapter 8: Enzymes. *In Medical Biochemistry*

[2] Ryu J. Y., Lee S. Y. 2019. *Proc. Natl. Acadm. Sci. U.S.A.* 116(28):13996–14001. https://doi.org/10.1073/pnas.1821905116.

[3] Tipton K., Boyce S. 2000. *Bioinformatics.* 16(1):34–40. https://doi.org/10.1093/bioinformatics/16.1.34.

[4] https://academic.oup.com/view-large/figure/389817290/gkac1052fig1.jpg

[5] Altschul S., Lipman D. 1990. *Journal of Molecular Biology.* 215(3), 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2.

[6] Dalkiran A., Doğan T. 2018. *BMC Bioinformatics*. 19:334. https://doi.org/10.1186/s12859-018-2368-y.

[7] Yu T., Zhao H. 2023. *Science*. 379, 1358–1363. https://doi.org/10.1126/science.adf2465.

[8] Elnaggar A., Rost B. 2022. *IEEE Trans Pattern Anal Mach Intell.* 44(10):7112-7127. https://doi.org/10.1109/TPAMI.2021.3095381.

[9] Paszke A., Desmaison A. 2019. *Advances in neural information processing systems.* 32. https://doi.org/10.48550/arXiv.1912.01703.

[10] https://github.com/peymanvahidi/pymmseqs

[11] Steinegger M., Söding J. 2017. *Nature Biotechnology*. 35(11), 1026–1028. https://doi.org/10.1038/nbt.3988.

[12] Pedregosa F., Vanderplas J. 2011. *Journal of Machine Learning Research.* 1;12:2825-30.

[13] McKinney, W. 2010. *Proceedings of the 9th Python in Science Conference, Austin*. 56-61. https://doi.org/10.25080/Majora-92bf1922-00a.

[14] Cock P. J. A., de Hoon M. J. L. 2009. *Bioinformatics*. 25(11):1422–1423. https://doi.org/10.1093/bioinformatics/btp163.

[15] Chicco D., Jurman G. 2020. *BMC Genomics*. 2;21(1):6. https://doi.org/10.1186/s12864-019-6413-7.

[16] Opitz J. 2022. *Heidelberg University*. https://www.cl.uni-heidelberg.de/~opitz/pdf/metric_overview.pdf.

[17] The UniProt Consortium. 2025. *Nucleic Acids Research*. 53(D1), D609–D617. https://doi.org/10.1093/nar/gkae1010.

[18] https://huggingface.co/Rostlab/prot_t5_xl_uniref50.

[19] Price M. N., Deutschbauer A. M. 2018. *Nature. 557(7706), 503–*

509. https://doi.org/10.1038/s41586-018-0124-0.
[20] Akiba T., Koyama M. 2019. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631. https://doi.org/10.1145/3292500.3330701.
[21] Biewald L. 2020. *Weights & Biases*. https://www.wandb.ai.
[22] Belkin M., Mandal S. 2019. *Proc. Natl. Acad. Sci. U.S.A*. 116 (32) 15849-15854. https://doi.org/10.1073/pnas.1903070116.
[23] https://github.com/tttianhao/CLEAN.
[24] Sanderson T., Colwell L. J. 2023. *Elife*. 12:e80942. https://doi.org/10.7554/eLife.80942.
[25] Davoudi S., Banaei-Kashani F. 2026. vbag004, https://doi.org/10.1093/bioadv/vbag004.