Enzyme Prediction

Machine Learning Models for Predictive Enzyme Classification

Malte Alexander Weyrich, Thanh Truc Bui, Jan Philipp Hummel and Sofiia Kozoriz

1 Abstract

The accurate prediction of enzyme commission (EC) numbers is not only crucial for the classification and understanding of newly discovered enzymes but also for completing the annotation of already known enzymes. Therefore, developing a reliable method for predicting EC numbers is very important.

However, due to insufficient data, enzyme function prediction using machine learning remains challenging. This paper proposes several methods for predicting enzymes in three different problem categories. Each category goes further into depth of the enzyme classification system, starting with a binary classification (Level 0) of enzymes and non-enzymes, followed by a multi-class classification (Level 1 & 2) of enzymes into main classes (e.g., EC X.-.-.-) and subclasses (e.g., EC X.X.-.-). Throughout developing our models, we used various input features and machine learning algorithms, of which the best will be thoroughly reviewed in this paper (see Table 1).

Level	Description	Model	F1	MCC
0	Binary classification	Random Forest	0.98	0.75
1	Main class prediction	Feedforward Neuronal Network	0.95	0.94
2	Subclass prediction	Feedforward Neuronal Network	0.85	0.83

Table 1. A table showing the best-performing models per Level

2 Introduction

Enzymes are vital as biological catalysts, facilitating essential biochemical reactions in organisms. Without enzyme catalysis, these reactions would occur slowly or practically impossible. Predominantly, proteins and enzymes are designated specific functions; for instance, oxidoreductases drive Redox reactions, and isomerases convert molecules into their isomers. Therefore, a comprehensive understanding and a precise classification of enzymes are fundamental.

In the past, scientists attempted to categorize enzymes into groups and develop a logical rule set for naming. The efforts were not successful due to ambiguity. A significant milestone occurred in 1956 with the establishment of an official international commission on enzyme classification (International Union of Biochemistry and Molecular Biology. Nomenclature Committee (1993)). This marked the initiation of the contemporary enzyme classification system, forming the basis for our understanding of enzymes today. In modern research, computational methods became invaluable for enzyme classification transforming the traditionally labor-intensive process. Machine learning and data-driven models assumed a leading role, holding the potential for enhanced accuracy and efficiency in annotating enzymes within vast genomic data. Several effective methods were developed to predict enzyme functions without the need for experimental validation. One notable example is DeepEC (Ryu *et al.* (2019)), a deep learning-based computational framework designed for the precise and high-throughput prediction of EC numbers for protein sequences. DeepEC employs three convolutional neural networks (CNNs) as a primary engine for EC number prediction and incorporates homology analysis for cases not classified by the CNNs.

Another noteworthy method is CLEAN (Yu *et al.* (2023)), which stands for “contrastive learning–enabled enzyme annotation”. CLEAN adopts a unique training objective to learn an embedding space for enzymes where the Euclidean ($L2$) distance reflects functional similarities. In CLEAN’s task, sequences sharing the same EC number exhibit a small distance, while sequences with different EC numbers have a larger distance. The model was trained using contrastive losses with supervision for effective enzyme function prediction.

In our research, we developed several classification models for each Level, using a variety of machine-learning approaches. The best-performing models per Level will be thoroughly reviewed in this paper. The first is a binary classifier determining whether a given protein functions as an enzyme. The second and third models are multi-classifiers, categorizing enzymes into classes and subclasses based on their specific functionalities.

3 Methods

3.1 Binary Classification of Enzymes

In the course of assigning EC numbers to protein sequences, our initial step is to determine whether a given protein sequence is enzymatic or non-enzymatic (Level 0). To identify the most effective approach for this task, we conducted the training of the following binary classification models and compared their performance to select the optimal one:

(1) k-Nearest-Neighbors
(2) Random Forest
(3) Support Vector Machine

The chosen method for binary classification is detailed in the subsequent section, while section 6.1 outlines the methods that were not selected.

3.2 Multi-class Classification of Enzymes

We used a Feedforward Neural Network (FNN) for levels 1 and 2 to classify the enzymes into their main classes and subclasses. The FNN is an artificial neural network that uses a series of fully connected layers to extract features from the input data and classify it into different categories, thus making it a suitable choice for our multi-class classification task.

3.3 Training Data

The protein data sets were taken from the UniProt database and used in the CLEAN publication on Enzyme function prediction (see Yu *et al.* (2023)). The complete data pool of enzymes (called Split100 in Yu *et al.* (2023)) contains 224693 sequences. These sequences are separated into several smaller subsets (SplitX, $X \in \{10, 30, 50, 70\}$), where X represents the amount of similarity allowed between the sequences. For reference, Split30 only contains sequences that share at most 30% similarity. Since we want to reduce family overrepresentation, we used the Split30 dataset for our research, which contains 9186 enzymes. With this redundancy reduction, we want to prevent bias towards a specific family of enzymes. This subset was combined with a non-enzyme dataset holding 37347 sequences, resulting in 46533 sequences.

3.4 Data Preprocessing

We removed all sequences containing an ambiguous amino acid, such as 'U' or 'O', from the dataset, as well as sequences with a length longer than 1022 amino acids. This is due to the enzyme dataset only containing sequences with a maximum length of 1022 amino acids. The non-enzyme dataset, however, contains many sequences with a length of up to 18000 amino acids. By removing these sequences we reduce the imbalance between the enzyme and non-enzyme dataset while also preventing the model from learning patterns that are specific to the non-enzyme dataset.

Additionally, we made use of protein embeddings. Large language models for protein sequences have learned the patterns of amino acid sequences by being trained on many protein sequences. This allows us to infer embeddings for our amino acid sequences, which are numerical vectors that represent the amino acid sequences. For each sequence we inferred ESM-2 embeddings and ProtT5 embeddings using the ESM-2 model (see Lin *et al.* (2023)) and the ProtT5 model (see Elnaggar *et al.* (2021)) respectively.

3.4.1 Data Preprocessing for Level 0

As a Random Forest model relies on multiple decision trees and each decision tree requires various features, we opted to extract additional information from the protein sequence and the ESM-2 embeddings Lin *et al.* (2023).

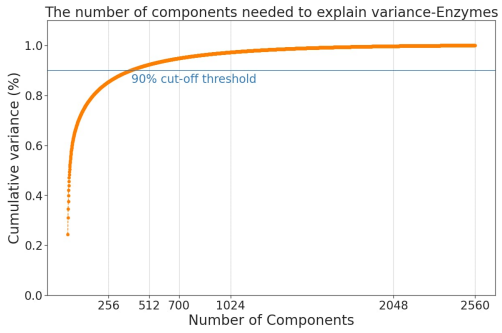

Fig. 1. The number of components needed to explain the variance in the enzyme dataset

From the protein sequences, we computed the mass by summing the individual masses of its amino acid components. Meanwhile, the ESM-2 embeddings underwent statistical analysis, where a 2560-dimensional vector represents each protein. We calculated the median (Emb. med.), standard deviation (Emb. std.), and vector magnitude (Emb. mag.) for each sample in the dataset. For clarity, the mass of protein sequences is presented in atomic mass unit (AMU), and the term 'vector magnitude' refers to the square root of the sum of the squares of all vector dimensions.

To simplify the embeddings, we applied Principal Component Analysis (Pedregosa *et al.* (2011)) separately to enzyme and non-enzyme datasets, aiming to retain 90% of the variance. Applying a 90% cut-off threshold resulted in reduced dimensions of 397 (see Figure 1) for enzymes and 369 for non-enzymes.

By standardizing both datasets to a dimensionality of 397, the higher dimensionality, we ensured that they remain at least 90% of the variance. This constant dimensionality of 397 was also employed to reduce the protein embeddings of unseen datasets during evaluation.

We also used the method SelectFromModel from the scikit-learn library (see Pedregosa *et al.* (2011)) to select features based on importance weights. The refined set of input features contains 'Mass', 'Emb. med.', 'Emb. std.', 'Emb. mag.', and 'PCA 1' through 'PCA 47' (see table 2) excluding 'PCA 29', 'PCA 31', 'PCA 35', 'PCA 36', 'PCA 39' to 'PCA 42', 'PCA 44' and 'PCA 46' (where 'Emb.' represents embeddings and 'PCA X' signifies the X-th dimension of the reduced embeddings).

Mass	Emb. med.	Emb. std.	Emb. mag.	PCA 1	…	PCA 47
60153.71	−0.00218	0.22733	11.50251	0.45033	…	−0.05049
81547.54	−0.00262	0.24014	12.15046	−0.08780	…	0.22034
…	…	…	…	…	…	…

Table 2. Input features for the Random Forest model.

3.4.2 Data Preprocessing for Level 1

For the Level 1 prediction, we labeled each sequence with its respective main class, resulting in 7 classes (see Figure 8). While the second main class is the most prevalent, the seventh main class is the least represented, with only containing 190 samples.

3.4.3 Data Preprocessing for Level 2

For the Level 2 prediction, we labeled each sequence with its respective subclass (see section 7 Figure 9). The problem of imbalanced data and underrepresentation of some labels is even more pronounced when labeling the sequences with their respective subclass. Some enzyme families

comprised less than ten representatives in the dataset, making it difficult for the model to learn the patterns of these families. To address this issue, we combined underrepresented families of the same main class into a single class (e.g. **EC 2.2.-.-**, **EC 2.9.-.-** and **EC 2.19.-.-** were merged into the label **EC 2.2|9|19.-.-**). This guaranties that each subclass has at least ten representatives in the dataset. This procedure had to be applied to each main class separately, as the subclasses are unique within their respective main class.

3.4.4 Training Procedure

We divided the data table into two parts for all our models: a training set and a validation set, using a random state of 42 for consistency. The training set contains 70% of the original data, while the validation set holds the remaining 30%. This separation allows us to train our models on a subset of the data and assess its performance on the validation dataset to ensure that the model does not overfit the training data.

3.4.5 Evaluation Procedure

The enzymatic dataset we used for evaluation is called "New-392" and was used as one of the benchmark datasets in the CLEAN publication (see Yu *et al.* (2023)). For the Level 0 prediction, we concatenated the "New-392" dataset with the non-enzymes test dataset. We also filtered for ambiguous amino acids and sequences longer than 1022 amino acids. The refined non-enzyme dataset contains 9336 non-enzymes, resulting in 9728 sequences for the Level 0 prediction. For the Level 1 and Level 2 predictions, we removed multifunctional enzymes that differ in their main class. This left us with 388 sequences for the Level 1 and Level 2 prediction. Non-enzymes were not included in the evaluation of the Level 1 and Level 2 predictions.

3.5 Model Architectures

3.5.1 Level 0 - Random Forest

A straightforward yet powerful machine learning method is the Random Forest algorithm. The choice of Random Forest is driven by its effectiveness in handling classification tasks, making it a well-suited option for our specific protein classification objective.

For the Random Forest classifier, we addressed the imbalance between the number of non-enzymes and enzymes, where non-enzymes outnumber enzymes approximately fourfold by replicating the enzyme data in the training set four times. This ensures a more balanced dataset, allowing the model to be trained on an equal representation of both classes.

We built a Random Forest Classifier using the scikit-learn library (see Pedregosa *et al.* (2011)) with specific parameters: The classifier consists of a total of 200 decision trees. Each tree has a maximum depth of 16 and a node is a leaf only if it has a minimum of 8 samples. The random state parameter is set to 42, ensuring reproducibility in the model's construction.

3.5.2 Level 1 - FNN

The FNN for Level 1, using ESM-2 embeddings, involves a sequential arrangement of linear layers, each accompanied by LeakyReLU activation functions. This choice of activation function, LeakyReLU, is particularly remarkable for its capability of dealing with the "dying ReLU" problem by allowing a small gradient when the unit is inactive, which we wanted to avoid. The architecture is as follows: The input dimension is 2560. The hidden layer node counts are 2048, 768 and 256. The output consists of 7 units. It is worth pointing out that the architecture contains batch normalization layers intended for normalizing the input to a layer through adjustment and scaling to stabilize and accelerate the training process. To address the imbalance in the dataset, we used class weights to penalize the model more for misclassifying the minority classes while being more lenient towards misclassifying the majority classes.

3.5.3 Level 2 - FNN

For the Level 2 prediction, we used a FNN to classify the enzymes into their respective subclasses. The architecture comprises one 2560 dimensional input layer , two hidden layers, and one 51 dimensional output layer, where 51 represents the number of subclasses in the training dataset (see Figure 9). The layers are fully connected, and the activation function used is ReLU. In between the hidden layers, we used dropout layers to prevent overfitting. The number of perceptrons used per hidden layer and the dropout rate were optimized using the Optuna library (see Akiba *et al.* (2019)). This resulted in 262 perceptrons in the first and 121 in the second hidden layer. The dropout rate was set to 0.23. During training, we used early stopping with a patience of 15 epochs to prevent overfitting. We also used class weights corresponding to the class frequencies in the Level 2 training dataset.

3.5.4 Scoring Metrics

$$\textbf{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\textbf{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\textbf{F1-score} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

$$\textbf{MCC score} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \cdot (\mathrm{TP} + \mathrm{FN}) \cdot (\mathrm{TN} + \mathrm{FP}) \cdot (\mathrm{TN} + \mathrm{FN})}}$$

To assess the performance of our models, we emphasize the F1 and MCC scores. Given the imbalance in the test set, we also considered the weighted F1 score for the multiclassification models to ensure a more equitable evaluation of the model's effectiveness. The F1 score is the harmonic mean of precision and recall, with values ranging from 0 to 1. A score of 1 signifies a flawless prediction, while 0 indicates an inaccurate prediction. The MCC score is a correlation coefficient that indicates the accuracy of binary classifications, with values ranging from -1 to 1. A score of 1 signifies a flawless prediction, -1 indicates a completely inaccurate prediction, and 0 suggests predictions at random. This scoring metric takes into account the entirety of the confusion matrix (TP, TN, FP, FN), making it a valid metric for measuring the performance across all classes. It can also be applied to multiclassification models by calculating the MCC score for each class and then averaging the scores.

3.5.5 Bootstrapping Procedure

Bootstrapping describes randomly drawing n samples with replacement from the same data pool B times. We used this method to further evaluate our model scores, as it allows us to estimate the mean, standard error, and confidence interval of the scores. We chose B as 10000, and n as the number of samples in the corresponding test set. This means that we created 10000 bootstrapped datasets, each of size n, containing the pairs of actual and predicted labels from the test set with replacement. This allowed us to calculate the scores for each bootstrapped dataset and then estimate the performance's mean, standard error, and confidence interval for each scoring metric. The scores were rounded to significant digits using the standard error. We calculated the error bars using an α of 0.05 and the standard error of the scores. The performance plots show the mean score and the confidence intervals as error bars (see Figure 5, 6 and 7).

4 Results

4.1 Binary Classification

Using the Random Forest model on the Level 0 test set, we achieved accurate predictions for about 90% of positive cases (enzymes) and 98% of negative cases (non-enzymes), even with an imbalanced test set comprising

392 enzymes and 9728 non-enzymes. As a reference, we provided the confusion matrix in Figure 2. Additionally, we attained an Accuracy of 0.97, a F1 score of 0.97, and an MCC-score of 0.74. Figure 5 visually demonstrates the considerable outperformance of our Random Forest model compared to the Random Baseline, which predicted labels based on the training set's proportions.

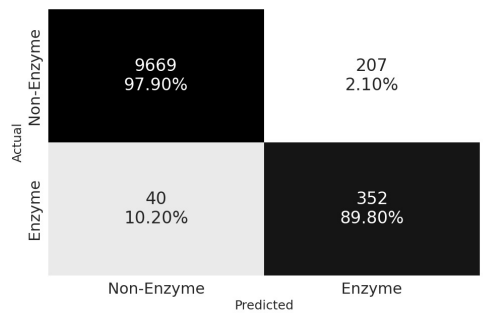

Fig. 2. Confusion Matrix of the Random Forest Model on the "new" dataset.

4.2 Main Class Prediction

After testing the FNN, we observed that the model achieved an F1 score of 0.95 and an MCC score of 0.94. The confusion matrix presented in Figure 3 demonstrates the model's robust predictive capability across seven classes, highlighting its high accuracy level. Most predictions align with the diagonal, and the percentages of accurate predictions stand out, ranging from 92.73% to 100% (see Figure 3). When comparing the FNN's performance to the Random Baseline, we observe that the FNN outperforms the Random Baseline in all metrics (see Figure 6). The FNN also managed to achieve slightly higher MCC score than CLEAN (0.92 versus 0.94) but overall performed similarly to CLEAN (see Figure 6).

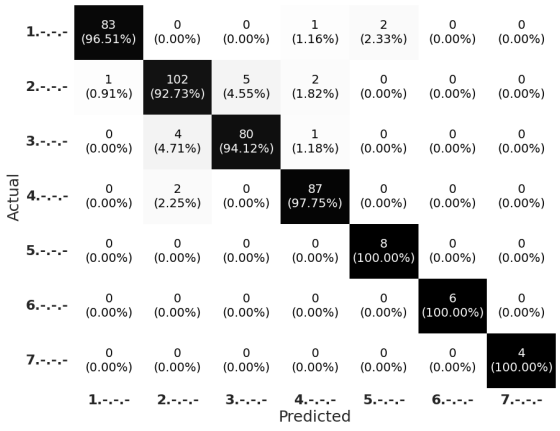

Fig. 3. Normalized confusion matrix of Level 1 prediction.

4.3 Subclass Prediction

The New-392 test dataset contains 28 out of the 51 in the training dataset. Our model achieved a weighted F1 score of 0.85 and an MCC score of 0.83 on the New-392 dataset (see Figure 7). We also calculated the confusion matrix of the model on the New-392 dataset (see Figure 4). The missing rows in the confusion matrix represent the 23 remaining subclasses that were not present in the New-392 dataset. The reasonably low noise per row indicates that the model can distinguish between the present subclasses of the New-392 dataset. Note that in Figure 4 the label 13 (corresponding to **EC 1.16.-.-**) seems to be wholly misclassified. This is because label 13 was the only representative of its class in the New-392 dataset and the FNN misclassified it as label 17 (corresponding to **EC 1.10|20|23|97.-.-**), making it seem like the model completely misclassified label 13. The Random Baseline for Level 2 achieved a weighted F1 score of 0.01 and an MCC score of 0.00 (see Figure 7).

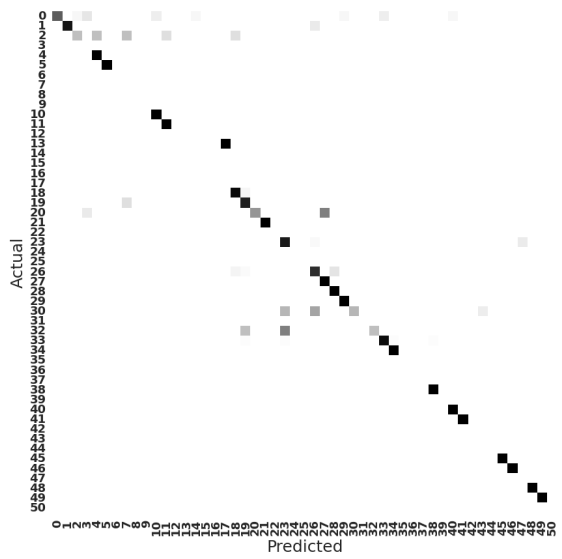

Fig. 4. Normalized confusion matrix of Level 2 prediction.

The level 2 FNN did not outperform CLEAN, which achieved a weighted F1 score of 0.90 and an MCC score of 0.88 (see Figure 7).

5 Discussion

The two main approaches we employed for the problem, namely Random Forest and Feedforward Neural Network, delivered promising results, surpassing the Random Baseline models. However, there is room for improvement due to the dataset's imbalance. One potential enhancement involves exploring additional features, such as incorporating information from protein 3D structures and protein interactions. These aspects could provide valuable insights and contribute to further refining the accuracy of our models. For the Random Forest model, where we applied a 90% cut-off for PCA, experimenting with a higher percentage, like 95%, could potentially enhance performance. We used class weights for both of our FNNs to address the imbalance in the datasets. However, this approach could be better, as it does not address the inability of the models to learn the patterns of the underrepresented classes. Instead of using class weights, oversampling the underrepresented classes could be a better approach to this problem. By oversampling, we aim to augment the representation of minority classes in the training dataset, providing the model with more instances to learn from. This approach can potentially enhance the model's ability to discern and generalize patterns related to the underrepresented classes, thereby improving its overall performance.

An idea for further development is to create a comprehensive pipeline by combining all three models to predict proteins from Level 0 to Level 2. In this hypothetical pipeline, the Random Forest model would predict Level 0. If the protein is identified as an enzyme, the Feedforward Neural Network (FNN) at Level 1 will predict the main class, while the other FNN at Level 2 will predict the subclass. Since the FNN at Level 2 implicitly

predicts the main class, we could align the predictions of the two models. If the main class predicted by the FNN at Level 2 matches the main class predicted by the FNN at Level 1, we would consider the entire result from the FNN at Level 2 as our final prediction. Otherwise, we would discard the prediction of the FNN at Level 2.

In conclusion, the three models represent promising tools for supporting researchers in classifying proteins. While each model demonstrates efficacy individually, the envisioned pipeline integrating all three could enhance protein classification's overall accuracy and reliability. Further exploration and implementation of these models may contribute valuable insights to the field of protein classification.

6 Supplementary Information

6.1 Additional Binary Classification Models

As we thoroughly reviewed the Random Forest model in section 4.1, we will outline the other binary classification models we considered for this task.

6.1.1 k-Nearest-Neighbors (kNN)

kNN is a non-parametric classification method that assigns a new object to the most common class amongst the most similar k objects in the data set (Hand (2007)). We implemented three kNN models in Python using scikit-learns Nearest Neighbors library, and these models used two distinct types of input features:

(1) Protein embedding vectors encoded by ProtT5 (see Elnaggar *et al.* (2021))
(2) Protein embedding vectors encoded by ESM-2 (see Lin *et al.* (2023))

Figure 5 illustrates the performance comparison of the kNN algorithm using ProtT5 and ESM-2 encoded embedding vectors separately, as well as the Random Baseline for Level 0. While both kNNs outperformed the Random Baseline, we observe that when using ProtT5 embeddings as input features, the mean F1 score slightly improved over the ESM-2 approach, recording 0.86 compared to 0.84. The same trend can be seen in the MCC score: 0.32, opposing 0.28. This implies that using ProtT5 as input vectors for the kNN approach showcases superior performance across both scores, especially the MCC score, the more robust measurement for imbalanced datasets.

6.1.2 Support Vector Machine (SVM)

SVM is an algorithm that assigns labels to objects by learning from examples (Boser *et al.* (1992)). We implemented two SVM models in Python using scikit-learns C-Support Vector classification library and these models using two distinct types of input features:

(1) Protein embedding vectors encoded by ProtT5 (Elnaggar *et al.* (2021))
(2) Protein embedding vectors encoded by ESM-2 (Lin *et al.* (2023))

Figure 5 illustrates the performance comparison of the two SVM models using ProtT5 embedding vectors and ESM-2 embedding vectors separately versus a Random Baseline. We observe that the mean F1 score for the ProtT5-based model is 0.93, and the mean F1 score for the ESM-2-based model is 0.93 compared to 0.84 for the Random Baseline. This implies that both SVM models perform with superior precision and recall compared to the Random Baseline. The MCC of 0.47 (ProtT5) and 0.49 (ESM-2), respectively, underline that both models perform better than random guessing.

6.2 Additional Multi-class Classification Models for Level 1

The following sections detail the architecture and results of further neural network models we created for Level 1 predictions.

6.2.1 FNN Model using ProtT5 Embedding

This FNN model was built similarly to our level 1 multi-class FNN 3.5.2. However, it uses ProtT5 embedding instead of ESM-2. Therefore, the layer sizes were adjusted. The model's architecture consists of an input layer of 1024 and three hidden layers of 1024, 512, and 7 units. The model also uses batch normalization between the layers, activation function LeakyReLU, class weights, and Adam as an optimizer. This model, however, performs worse than the FNN using ESM-2 embeddings.

This FNN model was built similarly to our Level 1 multi-class FNN in section 3.5.2. However, it uses ProtT5 embedding instead of ESM-2. Therefore, the layer sizes were adjusted. The model's architecture consists of an input layer of 1024 and three hidden layers of 1024, 512, and 7 units. The model also uses batch normalization between the layers, activation function LeakyReLU, class weights, and Adam as an optimizer. This model, however, performs worse than the FNN using ESM-2 embeddings.

6.2.2 One versus All Binary FNN Models

This model introduces an architecture of 7 binary FNNs, with each four densely connected layers for a binary classification problem of enzyme classes. It predicts each main class in a "one versus all" scenario. The first layer has ESM-2 embeddings of size 2560 as the input after it has the following layer dimensions: 2560, 1760, 1000, 500, 1. For ProtT5 with a tensor having 1024 inputs, the layers after the input layer were set to 512, 256, and 1. Each layer uses the ReLU activation function. The final output is passed through a Sigmoid activation function. In terms of performance, the model exhibits varied performances between embeddings, a strong indicator that although the architecture is static across different embeddings, it reacts differently to the distinct features of each embedding. On the validation dataset, the ESM-2 binary FNN had an F1 score of 0.70, and the ProtT5 model achieved 0.77. We did not test this model due to initial architectural mistakes, low performance, and the availability of better models.

Moreover, regularization techniques like dropout can mitigate the overfitting problem experienced in networks of this nature. To increase the performance, batch normalization, using a hyperparameter optimization framework, and optimizer Adam would be the possible future steps. Due to a lack of time and the complexity of testing it on unseen data, we did not see the potential of this model and moved on to the following architectures.

6.2.3 Convolutional Neural Network (CNN) Models

Before using ESM-2 embeddings, we must make them suitable for particular sequential data processing. The CNN model starts with a 1D convolutional layer that takes a single-channel input and applies 4 filters with a kernel size 61, moving with a stride of 1 and no padding. The output is then reshaped to a 4-channel 50×50 tensor. Following this, a 2D convolutional layer and 16 output channels uses a 3×3 kernel and a stride of 1 without padding, reducing the tensor size to 48×48. Max pooling with a 2×2 window and stride of 2 is applied, halving the dimensions to 24×24. Another convolutional layer with 16 input and 32 output channels maintains the size using a 3×3 kernel, stride of 1, and padding of 1. The subsequent pooling reduces it to 12×12. The process repeats with the third convolutional layer, increasing the channels to 64 while keeping the size constant, followed by pooling that reduces it to 6×6. The output is then flattened and passed through a fully connected layer mapping to 7 classes, with the Softmax function applied to the final layer's output to obtain class probabilities. We used ReLU activations after each convolutional layer for both CNN models. In the CNN model that

uses ProtT5 embeddings, the first layer is a 2D convolution with 1 input channel and 4 output channels, using a 3×3 kernel, stride of 1, and padding of 1, ensuring the output size remains the same as the input size. The next layer increases the channels to 16, followed by max pooling with a 2×2 window and stride of 2, reducing spatial dimensions by half. This pattern continues with the second convolutional layer increasing the channels to 32 and another pooling step, and then the third layer further increasing channels to 64 with a final pooling step. After the last pooling, the feature maps are flattened and passed through a fully connected layer with 1024 inputs (from the $6 \times 6 \times 64$ feature maps) to 7 outputs, with Softmax applied to the final output for class probabilities.

The ESM-2 CNN model performs better than the ProtT5 CNN model regarding both F1 and MCC scores. However, they are worse than the best FNN performance (see Figure 6).

In summary, each model reflects a different approach to the complex task of enzyme classification. While the models are promising, future directions include architecture optimization for different embeddings.

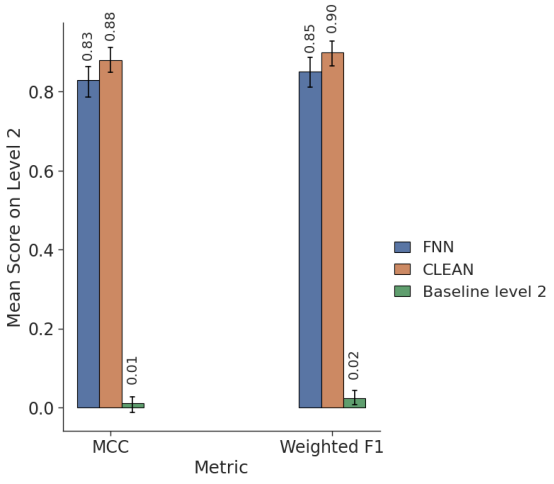

Fig. 7. Comparison between Level 2 FNN and CLEAN.

7 Supplementary Figures

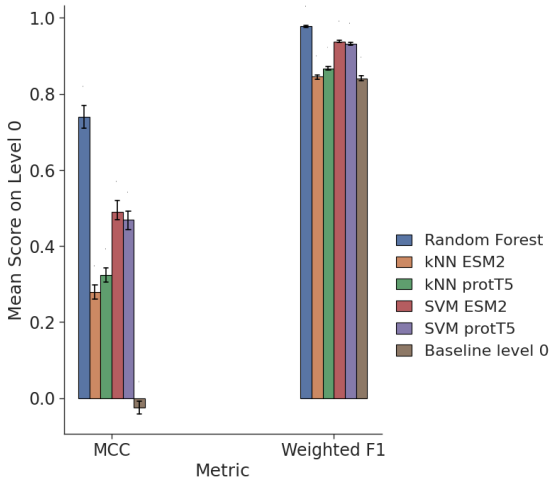

Fig. 5. Comparison of all Level 0 models.

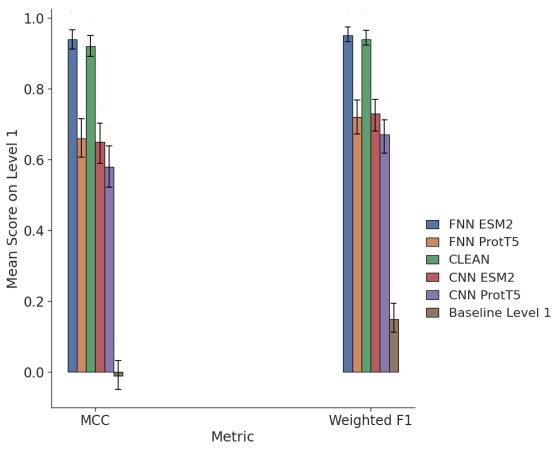

Fig. 6. Comparison of all Level 1 models.

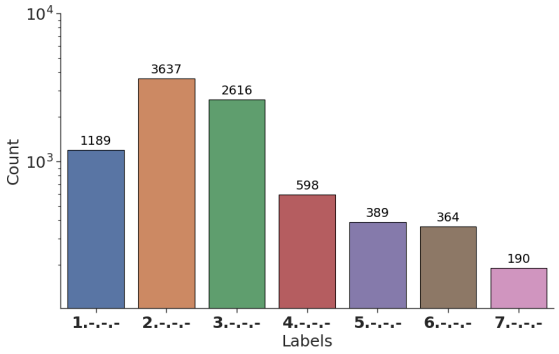

Fig. 8. Main class distribution of Split30 (logarithmic scale).

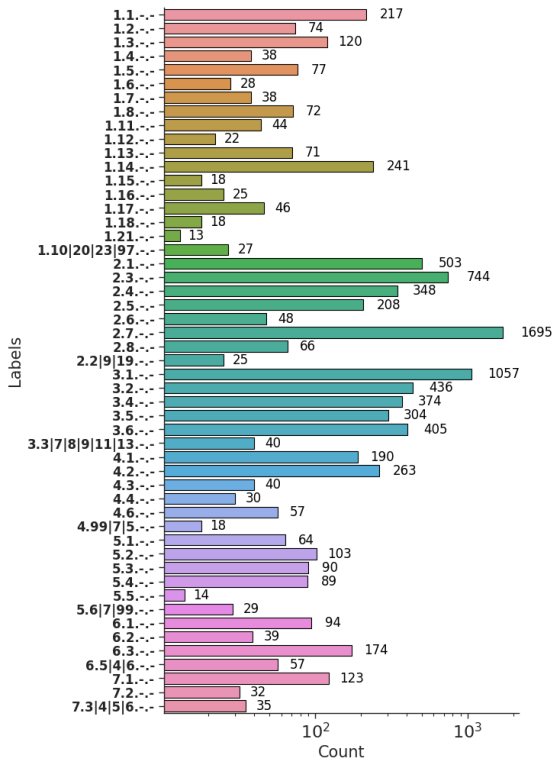

Fig. 9. Subclass distribution in Split30 (logarithmic scale).

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., *et al.* (2021). Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, **44**(10), 7112–7127.

Hand, D. J. (2007). Principles of data mining. *Drug safety*, **30**, 621–622.

International Union of Biochemistry and Molecular Biology. Nomenclature Committee (1993). *Enzyme nomenclature 1992*. Academic Press, San Diego, CA.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, **379**(6637), 1123–1130.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Ryu, J. Y., Kim, H. U., and Lee, S. Y. (2019). Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers. *Proceedings of the National Academy of Sciences*, **116**(28), 13996–14001.

Yu, T., Cui, H., Li, J. C., Luo, Y., Jiang, G., and Zhao, H. (2023). Enzyme function prediction using contrastive learning. *Science*, **379**(6639), 1358–1363.