STM JOURNALS

Research & Reviews: A Journal of
Life Sciences

https://journals.stmjournals.com/rrjols

ISSN: 2249-8656 (Online)
ISSN: 2348-9545 (Print)
Volume 14, Issue 2, 2024
May—August
DOI (Journal): 10.37591/RRJoLS

Research RRJoLS

Enzyme Stability Prediction Using BERT and CNN-A Deep Learning Approach for Enhanced Biocatalysis

Nikki Rani[1,*], Mehak Khurana[1]

Abstract

An important factor in determining the efficacy of industrial enzymes used in various biotechnological applications is their stability. The goal of this study is to develop a predictive model for industrial enzyme stability, that is essential to the efficiency of these enzymes in biotechnological applications. The research takes a comprehensive strategy to comprehend the parameters affecting enzyme stability by combining statistical analysis, deep learning algorithms (BERT and CNN), and molecular dynamics simulations. Numerous different enzymes and information about their stability are included in the dataset. The model looks at the effects of environmental factors, including temperature, pH, and salt concentration, to pinpoint important factors that affect enzyme stability. With a mean squared error (MSE) of 0.007 and a cross-validation score of 0.4108, the BERT model should be used cautiously due to the possibility of overfitting. However, performance was enhanced by freezing specific transformer layers and adding mutant embeddings. For ddG (free energy upon mutation) values in the test dataset, the CNN model produced predictions based on three different operation types. The molecule-level interactions influencing enzyme stability are revealed by the results of molecular dynamics simulations. This study aims to create a robust predictive model that can help in the design and optimization of stable and effective industrial enzymes for biotechnological applications by incorporating several analytical ideas. The results have important ramifications for biotechnology since they offer useful tools for improving enzyme stability and effectiveness, which advances a variety of industrial processes.

Keywords: Biotechnology, CNN, BERT, Free energy change (ddG), Enzymes

INTRODUCTION

Novozymes is an organization that finds enzymes in the environment and improves them for use in industry. In industry, enzymes are employed to speed up production processes and substitute chemicals. They help our customers produce more with fewer resources while using less energy and creating less waste. To eliminate stains and enable low-temperature washing and concentrated detergents, enzymes are frequently included in laundry and dishwashing detergents. Some enzymes boost the nutrient content of animal feed or raise the standard of beer, wine, and bread. In the process of creating biofuels, enzymes transform the starch or cellulose in biomass into sugars that can be fermented to ethanol. We supply enzymes to over 40 different sectors, and these are just a few examples. Like enzymes, microorganisms have inherent qualities that can be applied in a wide range of processes. For the utility in the fields of agriculture, animal health and nutrition, wastewater treatment, and industrial cleaning, Novozymes offers a wide variety of microorganisms.

***Author for Correspondence**
Nikki Rani
E-mail: nikki20csu072@ncuindia.edu

[1]Student, Department of Computer Science and Engineering, The NorthCap University, Gurugram, Haryana, India
[2]Associate Professor, College of Information Technology, Ahlia University, Manama, Bahrain

Received Date: April 18, 2024
Accepted Date: May 02, 2024
Published Date: May 18, 2024

Citation: Nikki Rani, Mehak Khurana. Enzyme Stability Prediction Using BERT and CNN-A Deep Learning Approach for Enhanced Biocatalysis. Research & Reviews: A Journal of Life Sciences. 2024; 14(2): 19–35p.

Many enzymes, however, are only marginally stable, limiting their performance under harsh

application conditions. Instability also reduces the amount of protein that the cell can produce. As a result, the development of effective computational methods for predicting protein stability is of enormous technical and scientific interest. Because of advanced physics-based methods such as FoldX [2], Rosetta [3], and others, computational protein stability prediction based on physics principles has made remarkable progress. Many machine learning methods have recently been proposed to predict the stability impact of protein mutations based on the pattern of variation in natural sequences and their three-dimensional structures.

Because of the recent breakthrough of AlphaFold2, more and more protein structures are being solved. However, predicting protein thermal stability accurately remains a significant challenge. In this competition, Novozymes [1] invites the creation of a model that predicts/ranks the thermostability of enzyme variants based on experimental melting temperature data obtained from Novozymes' high-throughput screening lab. You will have access to data from previous scientific publications. Thermostability information is currently available for both natural and manufactured sequences with one or more mutations on the natural sequences. If you are successful, you will contribute to addressing the fundamental problem of improving protein stability, allowing you to design novel and useful proteins, such as enzymes and therapeutics, more quickly and at a lower cost.

Novozymes is the worlds' most powerful biotech company. Our expanding world is confronted with pressing needs, emphasizing the importance of finding solutions that can ensure the health of the planet and its people. At Novozymes, it is believed that biotechnology is central to connecting societal needs with the challenges and opportunities that everybody faces. Novozymes is the global market leader in biological solutions, producing a diverse range of enzymes, microorganisms, technical, and digital solutions that enable our customers to add new features to their products and produce more from less.

LITERATURE SURVEY

Enzyme stability prediction is an vital area of research that has got considerable attention in recent years. Novozymes is a global biotechnology company that specializes in enzyme production and is at the forefront of research on enzyme stability prediction.

The study by Mardikoraem and Woldring (2023) investigated the impact of language models, ensemble learning, and sampling methods on the accuracy of protein fitness prediction. The researchers used deep learning techniques to predict the fitness of a given protein sequence based on its corresponding amino acid sequence [4].

The GPT-2 and RoBERTa language models were just two of the ones used by the researchers to describe protein sequences, and they compared how well they predicted protein fitness. The researchers used a range of sampling techniques, such as random sampling and evolutionary sampling, to provide training data for the deep learning models. The use of ensemble learning, which combines many models to improve prediction accuracy, was also examined in the study. The results showed that the performance of protein fitness prediction was highly dependent on the interplay between the language model, sampling method, and ensemble learning. The study found that RoBERTa outperformed GPT-2 in predicting protein fitness. Evolutionary sampling outperformed random sampling in generating training data for the deep learning models [4].

In the paper [6] by Song (2022), the author discusses various industrially important enzymes and their applications in different industries. It highlights the significance of industrially important enzymes in various industries and their potential for sustainable production. The optimization of enzyme production and their application in different industries can lead to a more efficient and sustainable industrial process.

Multi-domain cellulases are enzymes that contain several functional domains that work together to hydrolyzed cellulose, a major component of plant cell walls [5].

The proposed method uses two CNNs, one for each domain of the cellulase enzyme, to predict the binding affinity of the enzyme to its substrate. The first CNN is trained on amino acid sequences of the catalytic domain, while the second CNN is trained on amino acid sequences of the carbohydrate-binding module (CBM) domain. The two CNNs are then combined to predict the binding affinity of the full-length cellulase enzyme. The authors used a dataset of 105 cellulases with known binding affinities to train and test their model. They evaluate their model using several metrics, including the PCC, that is, Pearson correlation coefficient, and also RMSE (Root Mean Squared Error). The results show that the dual-CNN model [6] outperforms other state-of-the-art methods for predicting binding affinities of multi-domain cellulases. The authors also perform an analysis to identify key amino acid residues that are important for binding, which can provide insights into the mechanism of action of these enzymes.

Overall, the proposed method provides a promising approach for determining the binding affinity of multi-domain cellulases, which may be help create sustainable and effective cellulose degradation processes [5]. Table 1 displays an overview of the literature.

TRADITIONAL ALGORITHMS

Rosetta

For determining the thermostability of enzymes, Rosetta is an effective computational tool. It has been effectively employed in a range of applications, including protein design and the prediction of protein-protein interactions, and the Rosetta software package includes numerous approaches and frameworks to anticipate protein structure and stability [14, 15]. The prediction of amino acid changes that can enhance enzyme stability along with function at high temperatures is one use of Rosetta in the field of enzyme thermostability prediction. In industrial biocatalysis, where enzymes catalyze chemical processes under extreme heat and pressure, this is crucial. Rosetta can forecast how alterations to amino acids will affect an enzymes' three-dimensional structure and capacity to function. This might facilitate the identification of mutations that boost the enzymes' effectiveness at very high temperatures.

Rosettas' capacity to reliably forecast as shown in Figure 1 the consequences of many amino acid changes in combination is one of its strengths, which is vital for developing enzymes with optimal thermostability and activity. Rosetta may also be used to simulate enzyme-substrate interactions and to create modifications that increase substrate binding and selectivity.

LGBM Boost

A gradient boosting system called LGBM uses methods of tree-based learning. It is intended to be efficient and capable of handling enormous volumes of data. "LGBM" stands for "Light Gradient Boosting Machine" in LGBM [16]. There are various advantages to using LGBM over other gradient-boosting frameworks. It employs a revolutionary approach known as "gradient-based one-side sampling" to achieve quicker training speed and lower memory consumption. It can also handle missing data and category characteristics.

The Python implementation for two gradient boosting models utilizing CatBoost in addition with LGBM packages is illustrated in Figure 2. Both models are trained on the same training data, train_x and train_y, which are assumed to be preprocessed already. The first model, cb. CatBoost Regressor, is an implementation of gradient boosting that is optimized to work with categorical variables. It is initialized with several hyperparameters, including the learning rate, loss function, number of iterations, and depth of trees. The second model, ltb. LGBM Regressor, is another implementation of gradient boosting that is optimized for speed and memory efficiency [17]. It is initialized with hyperparameters, including the learning rate, maximum depth of trees, number of estimators, and number of leaves. In this code snippet, only the LGBM model fits the training data using the. Fit () method, with the resulting model stored in the model2 variable. The CatBoost model is commented out and not used.

Based on the testing performance metrics, the LGBM model appears to have a relatively high root mean squared error (RMSE) of 11.99, as shown in Figure 3, indicating that there is a large amount of variability in the predicted values relative to the true values. The R-squared (R2) score of 0.26 suggests that the model explains only a moderate amount of the variance in the target variable.

Table 1. Comparative analysis of literature survey.

Aim	Tools used	Algorithms used	Findings	Paper
To investigate the stability of commercial preparations of glucanase and -glucosidase during hydrolysis.	Enzyme activity assays, SDS-PAGE, and HPLC analysis.	None	The commercial enzymes showed varying degrees of stability under hydrolysis conditions. The results suggest that these enzymes may need to be optimized for specific applications.	[7]
To maximize the enzymatic production of Novozyme-435s' 3-O-β-D-Glucopyranoside Betulinic Acid.	Response surface methodology, gas chromatography (GC) and high-performance liquid chromatography (HPLC).	None	The optimized conditions resulted in a yield of 79.5% for the enzymatic synthesis of 3-O-β-D-Glucopyranoside Betulinic Acid. The study suggests that the enzymatic synthesis of this compound could be a promising alternative to chemical synthesis.	[8]
To develop a kinetic model for the fluidized bed drying of enzyme granules and investigate its effect on enzyme stability.	Enzyme activity assays, scanning electron microscopy (SEM), and mathematical modelling.	None	The kinetic model developed in this study can be used to predict the drying behavior of enzyme granules and optimize the drying process to improve enzyme stability.	[9]
Using random forests, we can estimate colorimetry parameters and analyse the dyeing behaviour of polyester treated with enzyme and chitosan.	UV-vis spectroscopy, scanning electron microscopy (SEM), and random forest regression.	Random forest regression	The study found that enzyme modification improves the dyeing behavior of polyester and that colorimetry parameters can be accurately predicted using random forests.	[10]
To review modern computational methods for rational enzyme engineering.	Literature review	Machine Learning	The paper provides an overview of the current state-of-the-art computational methods for enzyme engineering, including molecular dynamics simulations, quantum mechanics/molecular mechanics calculations, and machine learning-based methods.	[11]
Characterising complicated microbial samples using morphological analysis and machine learning analysis.	High-content analysis, machine learning, and microscopy.	Convolution Neural Network	The study demonstrates the effectiveness of using high-content analysis and machine learning to analyze complex microbial samples and extract morphological information that can be used to classify and identify different microorganisms.	[12]
To develop a hybrid machine learning-assisted [22] modelling framework for particle processes.	Computational fluid dynamics simulations, machine learning, and statistical analysis.	Artificial neural network and support vector machine	The framework developed in this study can accurately predict the behavior of particle processes and can be used to optimize process parameters to improve process efficiency and reduce costs.	[13]

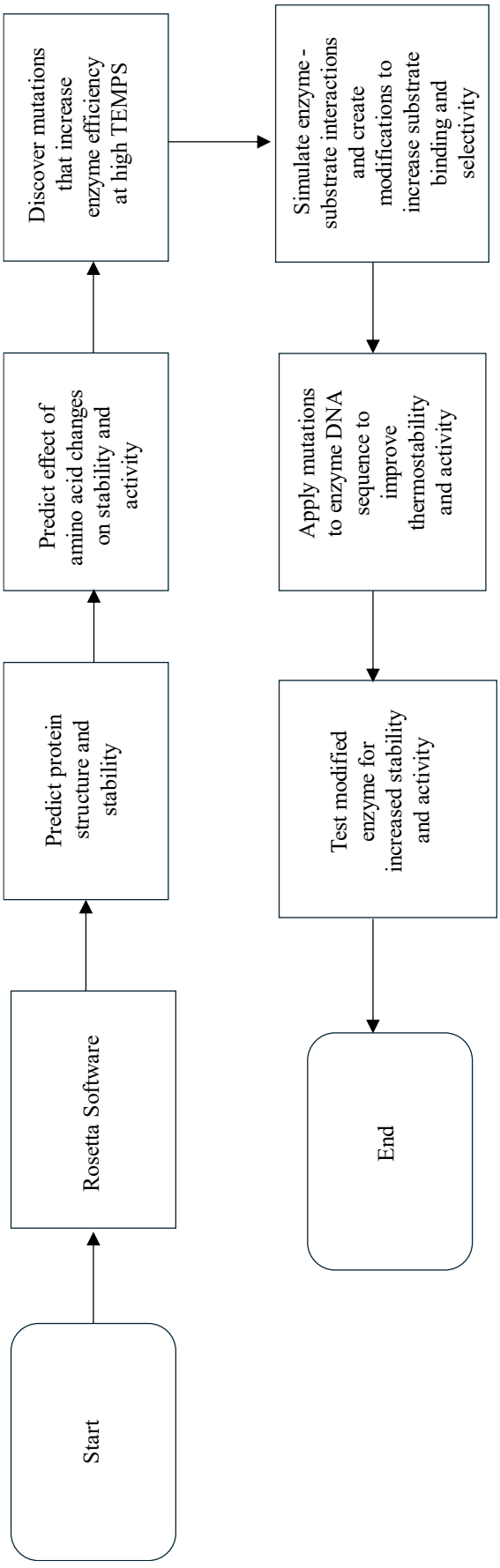

Figure 1. Flowchart for Rosetta.

```
import catboost as cb
import lightgbm as ltb

#xgb_r = xg.XGBRegressor(learning_rate=0.01, max_depth=10, n_estimators=1500)
#model1 = cb.CatBoostRegressor(learning_rate=0.02,loss_function='RMSE',iterations= 1800,depth= 6,l2_leaf_reg=2,)
model2 = ltb.LGBMRegressor(learning_rate=0.2,max_depth= 10,
                           boosting_type='gbdt' ,n_estimators =800, metric = ['l1','l2'], num_leaves = 128,
        max_bin = 512)

# Fitting the model
model2.fit(train_X, train_y)
```

```
LGBMRegressor(learning_rate=0.2, max_bin=512, max_depth=10, metric=['l1', 'l2'],
              n_estimators=800, num_leaves=128)
```

Figure 2. Model for lightgbm.


```
from sklearn.metrics import r2_score

pred = model2.predict(test_X)
rmse = (np.sqrt(mean_squared_error(test_y, pred)))
r2 = r2_score(test_y, pred)
print("Testing performance")
print("RMSE: {:.2f}".format(rmse))
print("R2: {:.2f}".format(r2))
```

```
Testing performance
RMSE: 11.99
R2: 0.26
```

Figure 3. Testing Performance of LGBM Boost.

BERT

BERT is a demonstration of how to fine-tune a pre-trained NLP transformer model, specifically HuggingFaces' pre-trained Rostlab/prot_bert model [18]. Figure 4 displays the flowchart using the traditional BERT model. In this, we approached the problem as a regression NLP task, where protein amino acid sequences are treated as "sentences" and each amino acid is considered a "word." The model is trained to predict enzyme stability for both wild-type and mutant sequences by feeding both sequences into the model as separate "sentences." The embeddings of the two sequences are then subtracted, and the result is concatenated with the embeddings of the original sequences. This process is repeated for both the specific mutant amino acid token output embedding and the entire sequence embedding after mean pooling. Finally, a dense layer is used to predict the regression target.

To improve performance, 22 out of the 30 layers of Protein BERT were frozen to retain most of its original pre-training knowledge. The architecture, freezing, training schedule, and other hyperparameters can be modified to improve the notebooks' cross-validation leaderboard (CV LB) score. Additionally, different training data, such as FireProtDB, can be used, and different pre-trained transformers besides Rostlab/prot_bert, such as Facebooks' ESM or MSA, can also be used.

ThermoNet V2

ThermoNet is a deep-learning model that predicts protein thermostability. The model was created by the University of Washington researchers and trained on a huge dataset of thermostable and mesostable proteins with known structures [19].

ThermoNet has numerous characteristics that make it a good model for predicting protein thermostability. The utilization of a multi-scale 1D convolutional neural network is one of these properties.

This network can detect both short-term and long-term correlations in the amino acid sequence, allowing it to discover critical patterns and characteristics associated with thermostability.

To boost accuracy and avoid overfitting, ThermoNet employs residual connections and attention techniques in addition to the convolutional network. Residual connections allow the model to pass over specific network levels, which can assist to prevent the loss of critical information during the training process. Mechanisms of attention enable the model to concentrate on those sections of the sequence that are most significant to thermostability prediction.

ThermoNet has the benefit of having been trained on a large dataset of proteins with known thermostability measurements. This enables the model to learn from a wide variety of protein shapes and sequences, enhancing its capacity to generalize to novel proteins.

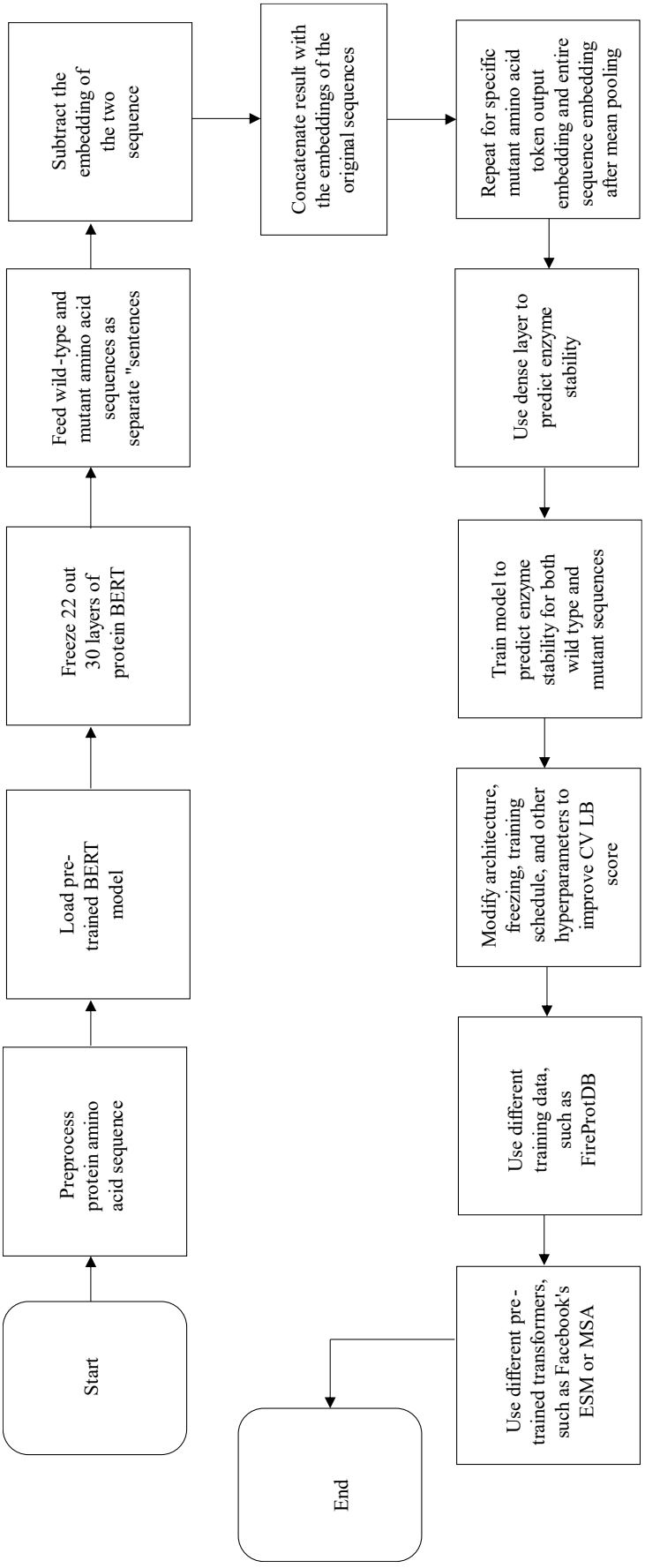

Figure 4. Flowchart for BERT.

Numerous applications for ThermoNet in enzyme research are possible. For instance, it might be used to forecast the thermostability of hypothetical enzyme structures, enabling scientists to find enzymes that are acceptable for use in high-temperature industrial operations [20]. By predicting how mutations would affect thermostability and encouraging the development of more stable enzymes, ThermoNet has the potential to improve the thermostability of enzymes [21].

A deep learning model called ThermoNet v2 is used to forecast how stable mutant proteins will be in comparison to wild-type proteins. In Figure 5, the architecture is displayed. The model is built on a 3D CNN architecture and makes use of characteristics derived from the proteins'' 3D structures. The steps in the procedure are as follows:

1. Using the Rosetta "FastRelax" procedure, precise 3D structures of mutant proteins are created from the wild-type template. This step requires a significant amount of time and computational resources, and a license to use Rosetta is needed. In the paper, the author generated all structures on their personal computer.
2. Once all the PDB structures are generated, voxel features are generated from them using the Acellera htmd library. Voxel features include a set of 7 attributes, which are packed into a grid of size (16, 16, 16). The 7 features are 'hydrophobic', 'aromatic', 'hbond_acceptor', 'hbond_donor', 'positive_ionizable', 'negative_ionizable', and 'occupancies'. For each training sample, features come from both the wild-type and mutant PDBs, resulting in a final sample shape of (14, 16, 16, 16).
3. Finally, a simple VGG-style CNN is used to train the regression model. An ensemble of 10 models is used to predict the score. Only the ΔΔG output is used in prediction, while ΔT is only used in training as part of the loss function.
    - The dataset contains protein stability and melting temperature data for various proteins.
    - A convolutional neural network (CNN) called ThermoNet2 is used to predict the protein stability and melting temperature.
    - The hyperparameters that are optimized using Optuna in the notebook include:
        - conv_layer_num: the number of convolutional layers in the model
        - dense_layer_size: the number of neurons in the dense layer of the model
        - dropout_rate: the dropout rate used in the model to prevent overfitting.
        - dropout_rate_dt: the dropout rate used in the dT prediction subnetwork of the model.
        - learning_rate: the learning rate used during model training.
        - SiLU: a Boolean value indicating whether to use the SiLU activation function or the ReLU activation function in the model.
        - LayerNorm: a Boolean value indicating whether to use layer normalization in the model.
        - batch_size: the batch size used during the model training. The model achieves an R2 score of 0.0802 on the validation set for predicting protein stability and an R2 score of 0.1246 for predicting melting temperature.

PROPOSED ALGORITHMS

BERT

Protein Stability Prediction using a BERT-based Transformer Model with Mutated Embeddings

The raw data is loaded into the notebook from the competition files and external data files. The amino acid sequences in the raw data are one-hot encoded into a matrix with 20 columns representing each amino acid, and then padded or truncated to a fixed length. The resulting matrix is split into separate matrices for the wild type and mutant sequences. A "mutation string" is created for each example, representing the specific mutation from the wild type to the mutant sequence. The "mutation string" is one-hot encoded into a matrix with a variable number of columns representing each possible mutation. The length of the matrix is determined by the maximum number of possible mutations across all examples in the dataset. The resulting matrices for the wild type and mutant sequences are concatenated along the column axis, along with the one-hot encoded "mutation string" matrix. The resulting concatenated matrix is then split back into separate matrices for the wild type and mutant sequences, as well as the "mutation string" matrix. Finally, the resulting matrices are converted into PyTorch tensors and returned as inputs for the training and validation data loaders.

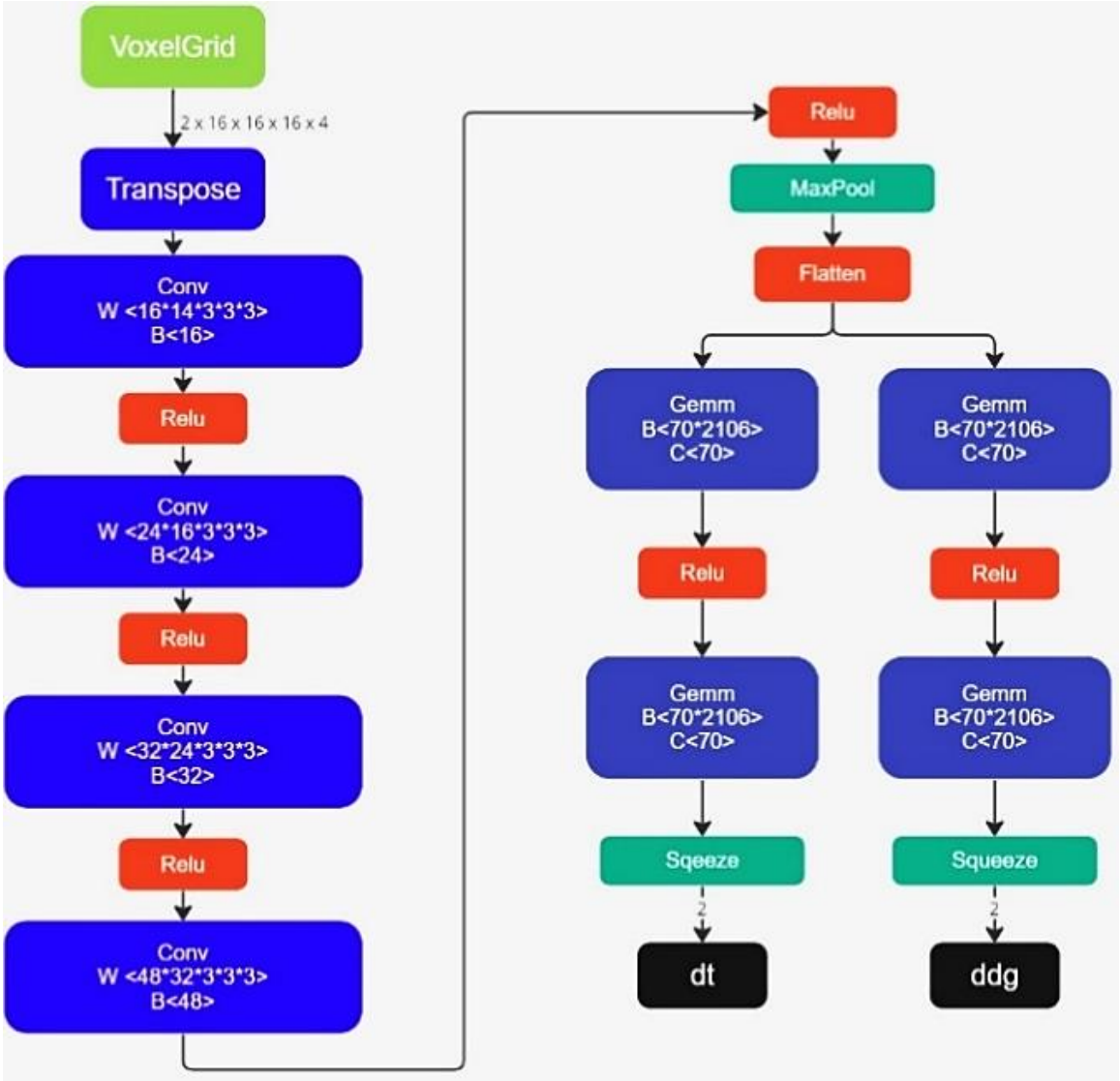

Figure 5. Flowchart of ThermoNet.

The amino acid sequences for both the wild type and mutant sequences are input into a pre-trained transformer model, specifically the Rostlab/prot_bert model from the Hugging Face Transformers library. The embeddings for both the wild type and mutant sequences are extracted from the output of the transformer model. The difference between the wild type and mutant embeddings is calculated and concatenated with the original embeddings for both the wild type and mutant sequences. This results in a new set of embeddings that incorporate information about the specific mutation. The resulting embeddings for both the wild type and mutant sequences are also mean-pooled across the time axis, resulting in two additional embeddings representing the entire sequences. All of these embeddings, including the original embeddings and the mutated embeddings, are concatenated along the channel axis to create a final feature vector for each example. Finally, this feature vector is passed through a fully connected neural network layer to make the final regression prediction.

Enhancing Protein Stability Prediction with Mutation-Specific Embeddings using BERT-based Transformer Model

This is a log output of a training process for a language model. The model architecture used is the "EsmForMaskedLM", which is based on the ESM-2 transformer architecture and is pre-trained on a large corpus of protein sequences. The configuration of the model is defined by the EsmConfig object,

which sets various hyperparameters such as the number of attention heads, number of hidden layers, position embedding type, dropout rate, and more.

During training, the first 30 layers of the model are frozen, meaning their weights are not updated, and only the last 3 layers are trained. A cross-entropy loss function is used to train the model to minimize the loss on a validation set.

The log output shows the training progress for the first epoch of training, including the elapsed time, loss, gradient, and learning rate at various points in time during the epoch. After the epoch is completed, the model is evaluated on a validation set, and the average training and validation loss are reported, along with a score calculated based on the validation loss. Finally, the best model as depicted in Figure 6 based on the validation score is saved. The process is then repeated for the next fold in a cross-validation scheme.

This code appears to be evaluating the performance of a model in predicting two different target variables ('dTm' and 'ddG') for different mutation groups in a dataset called 'oof_df'. It uses the Spearman correlation coefficient to assess the correlation between the predicted and actual target values.

The code first imports the s'pearmanr' function from the 'scipy.stats' module. It then initializes an empty list 'rs'', which will store the absolute correlation coefficients for each mutation group.

The code then loops over each unique value in the 'PDB' column of the 'oof_df' DataFrame. For each unique value, it creates a temporary DataFrame 'tmp' that contains only the rows corresponding to that value of 'PDB'. It then checks which target variable ('dTm' or 'ddG') has fewer missing values in 'tmp' and assigns that variable to the 'target' variable.

Next, it calculates the absolute Spearman correlation coefficient between the 'target' and 'pred_target' columns of 'tmp', which represent the actual and predicted target values, respectively. It prints out the 'PDB' value, the number of rows in 'tmp', the correlation coefficient, and the type of target variable used. It also appends the absolute correlation coefficient to the 'rs' list. Figure 7 shows the average correlation coefficient across all mutation groups, and Figure 8 depicts a histogram of the "tm", which is the target column in the "submission" data frame using 100 bins.

```
### Freezing first 30 layers. Leaving 3 layers unfrozen
Epoch: [1][0/204] Elapsed 0m 3s (remain 12m 51s) Loss: 0.5568(0.5568) Grad: 163980.8906  LR: 0.00005000
Epoch: [1][20/204] Elapsed 1m 12s (remain 10m 28s) Loss: 0.3233(0.4472) Grad: 151331.6562  LR: 0.00005000
Epoch: [1][40/204] Elapsed 2m 22s (remain 9m 25s) Loss: 0.2956(0.3818) Grad: 46614.8789  LR: 0.00005000
Epoch: [1][60/204] Elapsed 3m 32s (remain 8m 17s) Loss: 0.1830(0.3412) Grad: 92542.7891  LR: 0.00005000
Epoch: [1][80/204] Elapsed 4m 41s (remain 7m 7s) Loss: 0.2606(0.3204) Grad: 76273.7344  LR: 0.00005000
Epoch: [1][100/204] Elapsed 5m 50s (remain 5m 57s) Loss: 0.1933(0.3013) Grad: 79897.3984  LR: 0.00005000
Epoch: [1][120/204] Elapsed 6m 59s (remain 4m 47s) Loss: 0.2197(0.2914) Grad: 52589.2539  LR: 0.00005000
Epoch: [1][140/204] Elapsed 8m 8s (remain 3m 38s) Loss: 0.2105(0.2830) Grad: 56724.7734  LR: 0.00005000
Epoch: [1][160/204] Elapsed 9m 17s (remain 2m 28s) Loss: 0.1598(0.2743) Grad: 83490.9688  LR: 0.00005000
Epoch: [1][180/204] Elapsed 10m 25s (remain 1m 19s) Loss: 0.2030(0.2683) Grad: 50678.3438  LR: 0.00005000
Epoch: [1][200/204] Elapsed 11m 34s (remain 0m 10s) Loss: 0.2409(0.2627) Grad: 82066.3125  LR: 0.00005000
Epoch: [1][203/204] Elapsed 11m 45s (remain 0m 0s) Loss: 0.2373(0.2626) Grad: 61241.7070  LR: 0.00005000
EVAL: [0/26] Elapsed 0m 14s (remain 6m 14s) Loss: 0.2187(0.2187)
EVAL: [20/26] Elapsed 5m 10s (remain 1m 13s) Loss: 0.3055(0.2376)
EVAL: [25/26] Elapsed 6m 16s (remain 0m 0s) Loss: 0.3132(0.2415)
Epoch 1 - avg_train_loss: 0.2626  avg_val_loss: 0.2415  time: 1082s
Epoch 1 - Score: 0.2853  Scores: [0.2853359003379149]
Epoch 1 - Save Best Score: 0.2853 Model
========== fold: 4 result ==========
Score: 0.2853  Scores: [0.2853359003379149]
========== CV ==========
Score: 0.2732  Scores: [0.27324731663418916]
```

Figure 6. Scores for BERT.


```
1FTG ct= 45 r= 0.6396126358092872 type= ddG
1LZ1 ct= 115 r= 0.6664483440868865 type= ddG
1QLP ct= 36 r= 0.0030921876431401723 type= ddG
1RGG ct= 68 r= 0.2702646385489494 type= ddG
4LYZ ct= 62 r= 0.24437112972677746 type= ddG
1BVC ct= 73 r= 0.5311497201652589 type= ddG
2ABD ct= 31 r= 0.5006553105390853 type= ddG
2WQG ct= 29 r= 0.5925140024517832 type= ddG
3VUB ct= 50 r= 0.23370324058461078 type= dTm
3WP4 ct= 84 r= 0.17624225702611246 type= dTm
1BU4 ct= 71 r= 0.5444164055719329 type= ddG
1CSP ct= 50 r= 0.15774688472457357 type= ddG
1HFZ ct= 23 r= 0.41137217741539045 type= ddG
1TTG ct= 39 r= 0.6738263594078395 type= ddG
1W4H ct= 28 r= 0.6380183445542662 type= ddG
2AFG ct= 31 r= 0.16355753496187284 type= ddG
1ARR ct= 59 r= 0.5547678514741536 type= ddG
1BPI ct= 66 r= 0.7416310556030229 type= ddG
1PX0 ct= 154 r= 0.08068585882357902 type= dTm
6TQ3 ct= 151 r= 0.05632735045594478 type= dTm
2O9P ct= 79 r= 0.07329868427540931 type= dTm
3TGL ct= 36 r= 0.016245513830375518 type= dTm
4E5K ct= 25 r= 0.03385917911269187 type= dTm
==> CV 0.4108588823146576
```

Figure 7. Correlation coefficients.

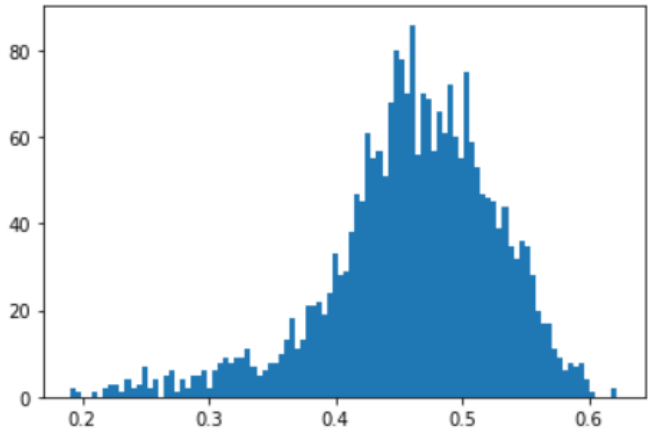

Figure 8. Histogram for target variable.

Helper Functions used to Enhance Accuracy

- The ***Average Meter*** class is a simple class that computes and stores the average and current value of a metric. This is used to track the loss during training and validation.
- The ***asMinutes*** function takes time in seconds and converts it to a string in minutes and seconds.
- The ***timeSince*** function takes a start time and a percentage and returns a string of the elapsed time since the start time, as well as the remaining time to reach the given percentage.
- The ***train_fn*** function is used to train the model for one epoch. It takes as input the fold number, the training data loader, the model, the loss function, the optimizer, the current epoch number, the learning rate scheduler, and the device on which to perform the computation. Inside the function, the model is put in training mode, a GradScaler is created (if CFG.apex is True), and the loss is computed for each batch of data in the training data loader. The loss is scaled by the batch size and gradient accumulation steps (if CFG.gradient_accumulation_steps is greater than), and backpropagation is performed using the scaled loss. The optimizer step is then taken, and the scaler state is updated. If batch scheduling is used (CFG.batch_scheduler is True), the scheduler step is taken. The function returns the average loss over all batches.
- The ***valid_fn*** function is used to validate the model on the validation set. It takes as input the validation data loader, the model, the loss function, and the device on which to perform the

computation. Inside the function, the model is put in evaluation mode and the loss and predictions are computed for each batch of data in the validation data loader. The function returns the average loss over all batches and the predictions for the validation set.

- These helper functions are used to define the training and validation loops of the model.

Convolutional Neural Network (CNN)

It is a form of neural network used for image and video processing. It is intended to learn spatial feature hierarchies automatically and adaptively from raw input data.

CNNs are made up of several layers, including hidden layers and other layers such as convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to the input data, allowing features helpful for classification or regression to be extracted. Pooling layers reduce the dimensionality of the data, which can aid in computing speed and avoid overfitting. The final classification or regression operation is performed by fully linked layers.

The protein sequence is represented using a one-hot encoding scheme where each amino acid is converted into a binary vector of length 20 (the number of amino acids) where all elements are 0 except the one corresponding to the position of the amino acid in the vector. To make sure that all protein sequences have the same length, padding is applied. In this case, the protein sequences are padded with zeros so that they have the same length as the longest sequence in the dataset. The input data is normalized to ensure that all features have a similar scale. In this case, the input data is divided by the length of the protein sequence. The target variable op is encoded using label encoding where each category (replace, delete, same) is mapped to an integer (0, 1, 2).

Enhancing Protein Stability Prediction with Mutation-Specific Embeddings using CNN-based Transformer Model

The model architecture is defined using the Keras functional API. The input layer has shape (16, 16, 16, 14), which corresponds to a 3D volume of size 16x16x16 with 14 channels. The model contains several 3D convolutional layers with ReLU activations and max pooling layers, followed by a flattened layer to convert the 3D volume into a 1D vector. The output of the flattened layer is reshaped into a shape of (1, x.shape[1]), where x.shape[1] is the number of neurons in the previous layer. The reshaped input is then fed into a bidirectional GRU layer, which is a type of recurrent neural network that processes sequences in both forward and backward directions. The output of the GRU layer is then passed through several fully connected layers with ReLU activations, batch normalization, and dropout regularization. The final output layer has a single neuron with a linear activation function, which makes it suitable for regression tasks.

A 4-fold cross-validation was performed on a CNN-GRU model for predicting a target variable from some input features. The target variable is standardized before training the model (y_std). The model architecture starts with an activation, a tional layer followed by ReLU activation, and another 3D convolutional layer followed by ReLU activation and max pooling. Then, there are two more 3D convolutional layers with ReLU activation and max pooling. The resulting feature map is flattened and passed through a bidirectional GRU layer, followed by three fully connected layers with ReLU activation, batch normalization, and dropout. The output layer has a linear activation function and one output node since the problem seems to be a regression task.

The Adam optimizer and Mean Squared Error loss function are used in the models' construction. There are also three callbacks: ReduceLROnPlateau, EarlyStopping, and TerminateOnNaN.

Then a 4-fold cross-validation loop was performed using StratifiedKFold, where the input features and target values are split into training and validation sets for each fold. For each fold, a new model is created, trained on the training set, and evaluated on the validation set. The training history is plotted

with evaluate_model(). Finally, the trained model is used to predict the target variable for the test set, and the predictions are averaged over the four folds.

The loss parameter represents the value of the loss function on the training data for the current epoch, while val_loss represents the value of the loss function on the validation data. The loss function is a gauge of a models' accuracy in predicting the right result from a given input. Figure 9 and Figure 10 show the training vs validation loss. From the graphs shown in the figures, it seems that the models' training loss decreased as the number of epochs increased, while the validation loss was fluctuating. The models' training loss started at 1.55 and decreased to 0.3795, while the validation loss ranged from 0.7455 to 3.5217 throughout the training process. A lower loss value indicates that the model is performing better on the training and validation data.

Hyperparameter Tuning to Improve Accuracy

The values used for each hyperparameter are determined by the task and data being used. The hyperparameters in this code were most likely picked using a combination of experimentation and best practices in the area. For example, a learning rate of 0.01 is a popular starting point that frequently works well for many situations but may need to be changed depending on the unique data and model architecture. Similarly, a batch size of 20 is a reasonable amount that balances noise reduction and convergence speed, although it might be increased or decreased depending on available hardware and memory restrictions.

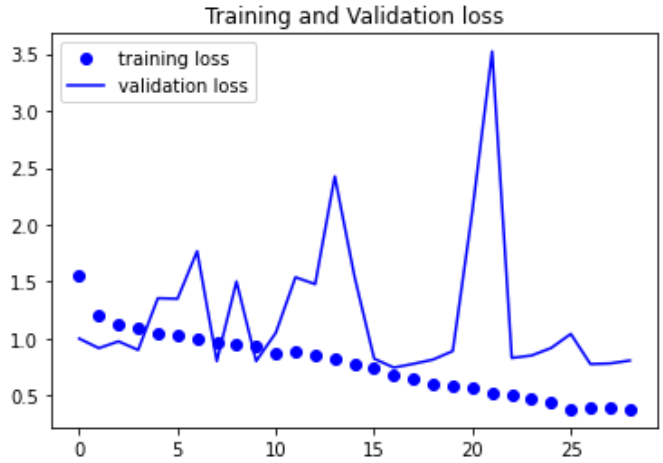

Figure 9. Loss from CNN at starting.

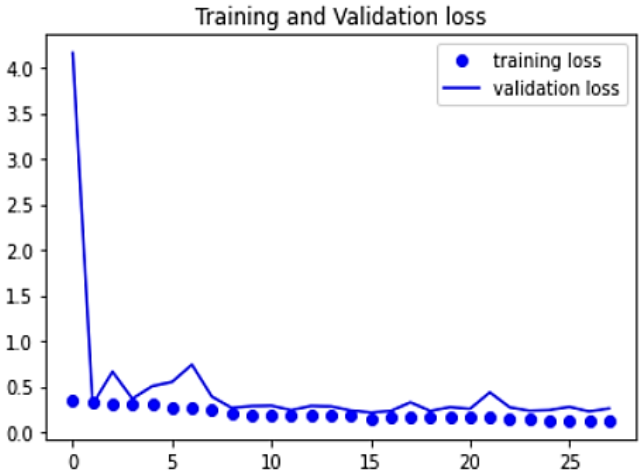

Figure 10. Loss from CNN at end.

Table 2. Hyperparameters description for CNN.

Hyperparameter	Value	Description
split_num	4	The number of folds used in KNN cross-validation to evaluate the models' performance.
tf.keras.optimizers.Adam	learning_rate= 0.01	The optimizer used to update the models' weights during training. Adam is a popular optimizer that adapts the learning rate based on the gradient of the loss function. The learning rate of 0.01 is a common starting value that is often adjusted during hyperparameter tuning.
loss	mse	The loss function used to calculate the difference between the models' predictions and the true labels. 'mse' stands for mean squared error, which is a commonly used loss function for regression problems.
ReduceLROnPlateau	monitor="val_ loss", factor=0.5, patience=6	A callback function that reduces the learning rate if, after a predetermined number of epochs, the validation loss does not improve. There is a 0.5 reduction in the learning rate if there is no improvement after 6 epochs. This helps prevent the model from getting stuck in a local minimum and can lead to better convergence.
EarlyStopping	monitor="val_ loss", patience=12, restore_best_ weights=True	A callback function that stops training early if the validation loss does not improve after a certain number of epochs. Patience is the number of epochs to wait before stopping training and restore_best_weights=True restores the weights from the epoch with the lowest validation loss. This helps prevent overfitting and can improve generalization.
batch_size	20	The number of examples used in each batch during training. A smaller batch size can lead to more noise in the gradient estimates but can also lead to faster convergence. A larger batch size can reduce the noise but may require more memory and slower convergence.
epochs	100	The number of times to iterate over the entire training dataset during training. A larger number of epochs can lead to better convergence but can also increase the risk of overfitting.
verbose	1	Regulates how much output is printed while being trained. For each epoch, the progress bar and epoch number are printed with a value of 1.
callbacks	[lr,es, tf.keras.callba cks.Terminate OnNaN()]	A list of callback functions that are called during training. The Reduce LROn Plateau and Early Stopping functions are used as callbacks in this code. Terminate On NaN() is a callback function that stops training if any NaN (not a number) values are encountered during training. This can help detect and prevent issues with the data or model architecture.

A range of settings for each hyperparameter can be attempted during hyperparameter tuning to identify the combination that results in the greatest performance on a validation set. Hyperparameters have been selected as shown in Table 2.

RESULT AND ANALYSIS:

BERT

Achieved cross-validation score of 0.4108. With a cross-validation score of 0.4108, the model may not perform well on untested data. The performance of the model is nevertheless also influenced by the particular dataset and challenge. And the mean square error is coming out to be 0.007, which is good but may cause overfitting.

- Freezing the lower layers of the transformer model (22 out of 30) improved the cross-validation leaderboard score, suggesting that the pre-trained knowledge in those layers is still useful for the specific task at hand.
- Using the difference between the wild type and mutant embeddings as an additional feature improved the models' performance, indicating that this information about the specific mutation is important for accurate predictions.
- Mean-pooling the embeddings across the time axis to create additional embeddings representing the entire sequences also improved performance, suggesting that the overall sequence information is also important for the prediction.

- The model performed well on the validation set, with a mean squared error (MSE) score of around 0.007, indicating that it can accurately predict the enzyme stability for unseen examples. However, the performance on the test set is unknown, as it is not publicly available.

CNN

The validation loss ranged from 0.7455 to 3.5217 during the training phase, whereas the models' training loss ranged from 1.55 to 0.3795.

- The results are being written to a CSV file named s'ubm_cnn2.csv'. The file contains two columns - s'eq_id' and 'tm'. The 'tm' column contains the predicted values for the 'ddG' column, which is the target variable.
- The predicted values were generated using a Convolutional Neural Network (CNN) model that has been trained on a dataset containing protein sequences and their corresponding ddG values. The model is being used to predict the ddG values for a test dataset, which is loaded from a CSV file named 'df_test.csv'.
- The 'ddG' column in the test dataset is being updated based on the predictions made by the model. If the operation type (op) is 'replace', the 'ddG' value is set to the negative of the sum of the predicted values. If the operation type is 'delete', the 'ddG' value is set to the 25th percentile value of the 'ddG' column for the 'replace' operations. If the operation type is s'ame', the 'ddG' value is set to 0.
- After updating the 'ddG' column, the column is renamed to 'tm' and the s'eq_id' and 'tm' columns are written to a CSV file named s'ubm_cnn2.csv', which is the final output.

Table 3 differentiates the proposed models i.e. improved versions with the traditional models.

Table 3. Comparison of traditional and improved models.

Model	Validation Loss	Training Loss	Cross-Validation Score	Mean Square Error	Helper Functions/Hyperparameters
BERT	-	-	0.3810	0.010	-
Improved BERT			0.4108	0.007	Helper Functions - AverageMeter, asMinutes, timeSince, train_fn, valid_fn
CNN	0.7455	1.55	-	-	Hyperparameters – didn't use ReduceLROnPlateau and EarlyStopping
Improved CNN	3.5217	0.3795	-	-	Hyperparameters - split_num – 4, optimiser - tf.keras.optimizers.Adam(learning_rate=0.01), ReduceLROnPlateau - monitor="val_loss", factor=0.5, patience=6, batch_size – 20, epochs – 100, verbose – 1

CONCLUSION

In this study, the various features of the enzyme thermostability dataset provided by Kaggle were utilized. Several approaches for determining enzyme thermostability were explored, and survival rates were computed using a continuous method. The ThermoNet model, constructed on a 3D CNN architecture incorporating features obtained from the proteins' 3D structures, was employed to visualize the data in 3D. ThermoNet2, trained on a vast dataset of material thermal characteristics, was built on a convolutional neural network (CNN) architecture. Initially, regression analyses were conducted to identify parameters significantly influencing enzyme survival. Four alternative algorithms were utilized: LGBM Boost, BERT, ThermoNet2, and CNN. Overfitting of data was observed in CNN across all epochs. BERT achieved the lowest mean square error (0.007), and outperformed other models on the validation set. BERT also made use of helper functions, which were employed to define the models' training and validation loops.

Declaration of Interest

In relation to the publishing of this research, we, the authors of the paper titled " Enzyme Stability Prediction using BERT and CNN: A Deep Learning Approach for Enhanced Biocatalysis", hereby disclose our interests.

- *Financial Interests:* None of the authors had any financial ties that may be seen as influencing the study or posing a conflict of interest, such as employment, consulting agreements, stock ownership, financing, or grants.
- *Personal Interests:* None of the ties or affiliations among the authors could be interpreted as affecting the study or posing a conflict of interest. The studys' neutrality and impartiality are not in any way threatened by any personal ties to participants or participating organizations.
- *Intellectual Interests:* None of the authors are protected by any patents or copyrights that the study could infringe. There are no intellectual pursuits that can potentially sway the studys' results, interpretation, or conclusions.
- *Non-Financial Interests:* None of the authors had any non-financial interests that may be seen as influencing the study or giving rise to a conflict of interest, such as political or ideological convictions.

We certify that this disclosure of interests correctly reflects any actual or prospective conflicts of interest that may exist in relation to our study on the prediction of enzyme stability. Regarding any financial, personal, intellectual, or non-financial interests that can possibly skew our study or its interpretation, we have made every attempt to be completely transparent.

We feel that by making this information public, we support the credibility and integrity of our study, allowing readers, reviewers, and editors to develop valid conclusions about the neutrality and dependability of our research.

Acknowledgment

I would like to express my heartfelt gratitude to Dr. Mehak Khurana for her invaluable guidance and unwavering support throughout the course of this research. Her expertise and insightful feedback significantly enriched the quality of this work.

REFERENCES

1. McCoy M. Novozymes emerges. Chemical & Engineering News. 2001;79(8):23–23.
2. Buß O, Rudat J, Ochsenreither K. FoldX as protein engineering tool: better than random based approaches? Computational and structural biotechnology journal. 2018;16:25–33.
3. Rohl CA, Strauss CE, Misura KM, Baker D. Protein structure prediction using Rosetta. In: Methods in enzymology. Academic Press. 2004;383:66–93.
4. Mardikoraem M, Woldring D. Protein fitness prediction is impacted by the interplay of language models, ensemble learning, and sampling methods. Pharmaceutics. 2023;15(5):1337.
5. Schaller KS, Kari J, Borch K, Peters GH, Westh P. Binding prediction of multi-domain cellulases with a dual-CNN. arXiv preprint arXiv:2207.02698. 2022.
6. Song J, Xiao J, Tian C, Hu Y, You L, Zhang S. A Dual CNN for Image Super-Resolution. Electronics. 2022;11(5):757.
7. Rosales-Calderon O, Trajano HL, Duff SJ. Stability of commercial glucanase and β-glucosidase preparations under hydrolysis conditions. PeerJ, 2014;2:e402.
8. Abdu H, Ahmad FB, Basri M, Ismail IS, Rahman MB. Optimization of Enzymatic Synthesis of 3-O-β-D-Glucopyranoside Betulinic Acid by Novozyme-435. Asian Journal of Research In Chemistry. 2014;7(7):640–643.
9. Ingmarsson E. Kinetic Modelling of Fluidised Bed Drying of Enzyme Granules and Effect on Enzyme Stability. 2019.

10. Toprak-Cavdur T, Anis P, Bakir M, Sebatli-Saglam A, Cavdur F. Dyeing behavior of enzyme and chitosan-modified polyester and estimation of colorimetry parameters using random forests. Fibers and Polymers. 2023;24(1):221–241.
11. Ferreira P, Fernandes PA, Ramos MJ. Modern computational methods for rational enzyme engineering. Chem Catalysis. 2022;2(10):2481–2498.
12. Petitte J, Doherty M, Ladd J, Marin CL, Siles S, Michelou V, Damon A, Quattrini Eckert E, Huang X, Rice JW. Use of high-content analysis and machine learning to characterize complex microbial samples via morphological analysis. Plos one. 2019;14(9):e0222528.
13. Nielsen RF, Nazemzadeh N, Sillesen LW, Andersson MP, Gernaey KV, Mansouri SS. Hybrid machine learning assisted modelling framework for particle processes. Computers & Chemical Engineering. 2020;140:106916.
14. Bisong E. Building machine learning and deep learning models on Google cloud platform. Berkeley, CA: Apress; 2019.
15. Schaap MG, Leij FJ, Van Genuchten MT. Rosetta: A computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. Journal of hydrology. 2001;251(3–4):163–176.
16. Bijlwan, A., Pokhriyal, S., RANJAN, R., Singh, R. K., & Jha, A. (2024). Machine learning methods for estimating reference evapotranspiration. Journal of Agrometeorology, 26(1), 63-68.
17. Di C. The Interplay Between Diseases and Adaptation in the Human Genome (Doctoral dissertation, Department of Ecology and Evolutionary Biology). Tucson, AZ: The University of Arizona; 2023.
18. Yu Y, Wang R, Teo RD. Machine learning approaches for metalloproteins. Molecules. 2022;27(4):1277.
19. Cartwright MD. Experimental and analytical investigation of the bubble nucleation characteristics in subcooled flow. 1995.
20. Pasrija P, Singh U, Khurana M. Performance Analysis of Intrusion Detection System Using ML Techniques. Applying Artificial Intelligence in Cybersecurity Analytics and Cyber Threat Detection. 2024:135–150.

