VOLUME 19, N° 1 2025
Journal of Automation, Mobile Robotics and Intelligent Systems

Implementation of Enzyme Family Classification by using Autoencoders in a Study Case with Imbalanced and Underrepresented Classes

Submitted: 15th April 2024; accepted: 20th May 2024

Darian Fernández Gutiérrez, Ariadna Arbolaez Espinosa, Deborah Raquel Galpert Cañizares, María Matilde García Lorenzo

DOI: 10.14313/jamris-2025-005

Abstract:
In the field of Bioinformatics, the scientific community is fully aware of the challenges associated with enzyme classification. In this study, a novel strategy is proposed based on the use of Anomalous Autoencoders to characterize chitinases belonging to glycoside hydrolases. Python and TensorFlow programming technologies were employed to conduct this analysis. The designed classifier consists of two levels that determine both the enzymatic nature of an amino acid sequence and its corresponding chitinase enzyme family. These levels considered class imbalance and the underrepresentation of those enzyme families in the CAZy.org database. Furthermore, a comprehensive comparison was made with other available software in the field. To represent the amino acid sequences, embeddings generated from the ProtFlash model were used. The results obtained in this study confirm the effectiveness of the proposed implementation compared to the methods EzyPred, ECPred, and Proteinfer.

Keywords: *autoencoders, bioinformatics, embeddings, enzyme classification*

1. Introduction

Elucidating protein or enzyme functions from amino acid sequences is still an open field in Bioinformatics although very outstanding solutions have been reported in literature including machine and deep learning approaches [1]. Trying to fit this general problem to the recognition of specific enzymes we detect a lack of data affecting model construction and validation. Our project involves the exploration of complete proteomes of microorganisms to identify potential chitinase enzymes based on the CAZy,org database, which contains glycoside hydrolases (GH) enzymes where chitinases are found within the GH18 and GH19 families. The complexity lies in the classification of sequences that are not very similar to the annotated (or labeled) ones, due to their low representativeness in several classes, or sequences that are similar to the annotated ones, but belong to a different class.

To address this, some methods have been developed that leverage various sources of information, such as different representations of k-mers [2, 3], and that "learn" from unlabeled sequences in a semi-supervised approach, although the accuracy in classification remains an open problem [4, 5].

Other classification methods rely on embedded representations that include information from previous classifications by pretraining them with databases of amino acid sequences [6–8]. The embeddings transform the sequences into numerical vectors through natural language processing (NLP) while incorporating heterogeneous data sources [5].

In addition, non-standard classification methods with a semi-supervised approach or one-class learning [9] have been tackling the low representativeness in the classes. Anomalous Autoencoders can be considered as a one-class learning approach because they are trained using only normal data [10, 11]. By exposing the model to anomalous data, the resulting reconstruction is expected to be significantly different from the normal data. This discrepancy can serve as a measure of the anomaly present in the input data [12].

On the other hand, class imbalance is a widely addressed problem in machine learning and has been considered in enzyme classification by incorporating information from labeled sequences into deep neural network models combined with oversampling methods like SMOTE [13].

The proposal of this work is to develop a classification method using embedded representations and Anomalous Autoencoders, considering unlabeled sequences or heterogeneous sources to achieve high efficacy in a study case of chitinases with low representativeness in the family classes, which are also imbalanced.

2. Content

In the training process of the Autoencoders [14] in this research, sequence data belonging to the GH18 and GH19 enzyme families were used (Table 1). These sequences were obtained from the CAZy.org database, which is recognized for its extensive information on enzymes related to carbohydrate degradation. All the training sequences used from the GH18 and GH19 families exhibit the same enzymatic activity with the EC number 3.2.1.14.

Open Access. © 2025 Darian Fernández Gutiérrez et al., published by Sciendo. BY-NC-ND This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License

Table 1. Number of enzymes per family

Family	Number of enzymes
GH18	356
GH19	83

To convert the enzyme sequences into numerical representations, the embedding technique proposed in the ProtFlash study was employed [15]. This technique allowed the generation of feature vectors of size 768, which capture relevant information from the sequences and facilitate their further processing in the context of machine learning workflows.

The embedded vectors can exhibit a wide variation in the values they can span, ranging from very distant ranges, such as -255 to 55. To address this diversity and improve the performance of neural networks, the Min-Max scaling technique was applied [16]. As shown in Table 1, there is an imbalance between classes of the families under study, and in general, low representativeness, so preprocessing to increase the training set could improve classification. Later on, an experimentation related to this is shown.

Subsequently, these resulting feature vectors were used as input in the training of the Autoencoders. In this way, a compact and meaningful representation of the enzyme sequences was obtained, which is essential for subsequent classification.

2.1. Classification of the Embeddings

In this study, a sequence classification approach for enzyme sequences is employed using a classifier based on Autoencoders with a multi-level structure. The main objective is to accurately identify and then categorize chitinase enzyme sequences.

In the first level of the classifier, an evaluation is performed to determine if a given sequence corresponds to an enzyme. If the presence of an enzyme is confirmed, the sequence is directed to the next level of the classifier. In this second level, the prediction of the enzyme family to which the studied sequence belongs is carried out. The flow of processes of the multi-level classifier used in this work is illustrated in Fig. 1.

Due to the limited representativeness of the available sequences for model construction, the SMOTE method [17–19] was employed. This approach allowed for the generation of synthetic data, resulting in improved outcomes as shown in Tables 2, 3, and 4. It is important to highlight that the number of sequences significantly increased from 439 to over 700 through this strategy.

2.1.1. First Level

In the initial stage of the classifier, an anomalous Autoencoder architecture is used for the identification of enzyme sequences. This Autoencoder has been previously trained using training sequences (embeddings) associated with the GH18 and GH19 families. The primary function of the Autoencoder is to determine whether a given sequence corresponds to an enzyme or not.

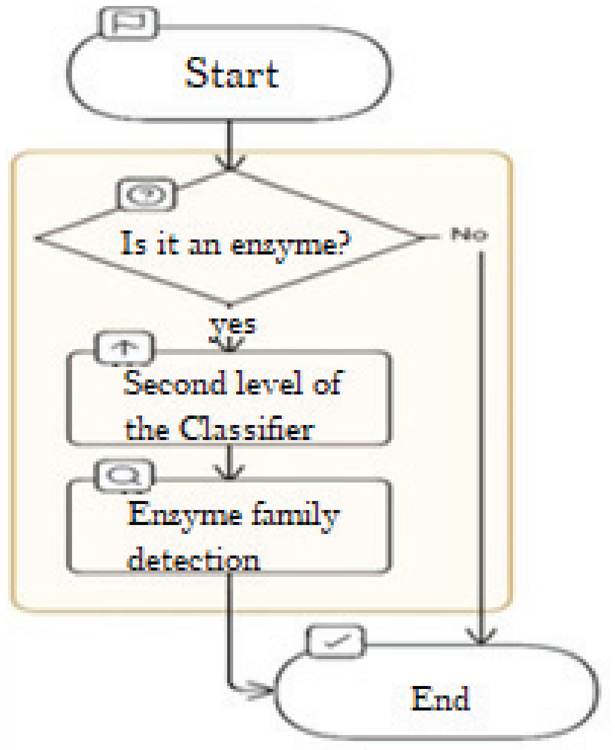

Figure 1. Autoencoder-based two-level classifier

To fulfill this purpose, a threshold is established as a classification criterion to determine whether a vector reconstructed by the Autoencoder corresponds to an enzyme or not. The calibration of this threshold is performed with the aim of achieving a classification accuracy level of 99%, using equation (1) to compare the reconstructed data with the training data.

$$distance(\mathrm{A},\ \mathrm{B}) = \frac{1}{n}\sum_{i=1}^{n}(A_i - B_i)^2 \qquad (1)$$

Where:

A_i: The corresponding element at position i of the reconstructed data A,
B_i: The corresponding element at position i of the training data B and
n: sequence length

The architecture of the Autoencoder consists of two parts: the encoder and the decoder. The encoder takes the original input sequence, which has a dimension of 768, and transforms it into a lower-dimensional latent representation through a series of dense layers with ReLU activation functions [14, 20]. Subsequently, the decoder performs the reverse process, reconstructing the original sequence from the latent representation.

During the training process, an exhaustive search is applied, which is a technique used in hyperparameter optimization to find the optimal combination of parameter values for a machine learning model. Different values are considered for the number of epochs (25, 50, 100, 150), batch size (32, 64, 128), optimizer (Adam, Nadam), and learning rate (0.001, 0.01, 0.1).

Table 2. Comparison of the trainings of the First Level

	Loss Function	Loss Function (Validation)
Without SMOTE	0.0202	0.0223
SMOTE	0.0127	0.0162
Hyperparameter optimization. (SMOTE)	<u>0.0025</u>	<u>0.0074</u>

The best values are underlined.

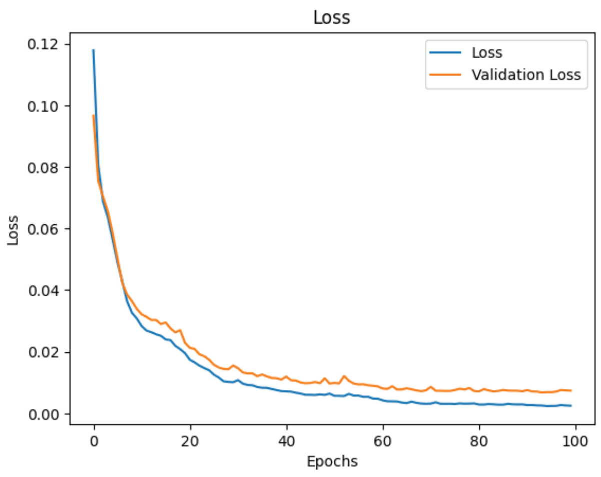

Figure 2. Loss function (MSE) of the first level classifier using SMOTE

Another strategy used during model training to prevent overfitting was early stopping, implemented through the "early_stopping" object. The validation loss is monitored, and if it does not improve after a certain number of epochs (in this case, 5), the training is stopped, and the model weights are restored to the best ones obtained. This ensures that the best model configuration is preserved and avoids unnecessary training.

After running the grid search with all possible combinations along with early stopping, the best values were found: a batch size of 32, 100 training epochs, a learning rate of 0.001, and the Adam optimizer [21]. In Fig. 2, the graphical representation of the loss function is shown, which corresponds to the mean squared error (MSE). It can be observed how this loss decreases throughout the 100 training epochs, indicating an improvement in the reconstruction of the autoencoder.

Table 2 shows the results obtained after training the first level, both before and after using SMOTE. It can be observed that there is an improved performance when using SMOTE.

2.1.2. Second Level

The second level of the classifier focuses on predicting the enzyme family to which each enzyme belongs. This process is divided into two stages: Stage 1 and Stage 2 [22].

In Stage 1, the training of Autoencoders is carried out individually, where each Autoencoder is exclusively trained with enzyme sequences related to a specific family. This process allows each Autoencoder to learn latent representations of sequences from its corresponding family.

In Stage 2, enzymes are classified into their respective families using the reconstruction losses calculated through the pre-trained Autoencoders (AE1, AE2, ..., AEn). In our case, only two Autoencoders were pre-trained, one for each family (GH18 and GH19). To calculate these losses, each enzyme sequence is passed through the pre-trained Autoencoders, and the reconstruction losses are computed for each Autoencoder. These losses are then used as inputs for a sigmoidal dense layer classifier [14, 20], where the corresponding enzyme families (F1, F2, ..., FN) are the outputs. In other words, the model is trained to predict the enzyme family based on the reconstruction losses generated by the Autoencoders (Fig. 3).

To achieve this, the weights of the pre-trained Autoencoders are set as non-trainable [22]. Then, a joint network is created that takes the input sequence and passes it through the two corresponding Autoencoders for the GH18 and GH19 families. The outputs of these Autoencoders are concatenated, and a dropout layer is applied [23] to regularize the model. Next, a dense layer with ReLU activation is employed to learn intermediate representations. Finally, a softmax activation is applied in the output layer for the classification into one of the two analyzed enzyme families. The final architecture of the two-level classifier can be seen in Fig. 4 once implemented using the TensorFlow library.

For the training process, the same hyperparameter optimization method as in Level 1 is used, along with early stopping. In this case, the best values were a batch size of 32, 50 training epochs, a learning rate of 0.1, and the Nadam optimizer [24]. The loss function used during this classification stage is the categorical cross-entropy, which is shown in Fig. 5 over 50 epochs. This graph allows us to observe the performance obtained during training, while the accuracy is used as the evaluation metric.

Table 3 shows the results obtained after training the second level, both before and after applying SMOTE. It can be observed that there is an improved performance when using SMOTE.

2.2. Results Analysis

To evaluate the results of the developed classifier, three existing methods for enzyme classification were examined, such as Ezypred [25], ECpred [26] and ProteInfer [27]. Although these methods generally exhibit satisfactory performance, this performance might not remain when dealing with sequences from imbalanced and poorly represented classes, since they do not address classification into such kinds of families.

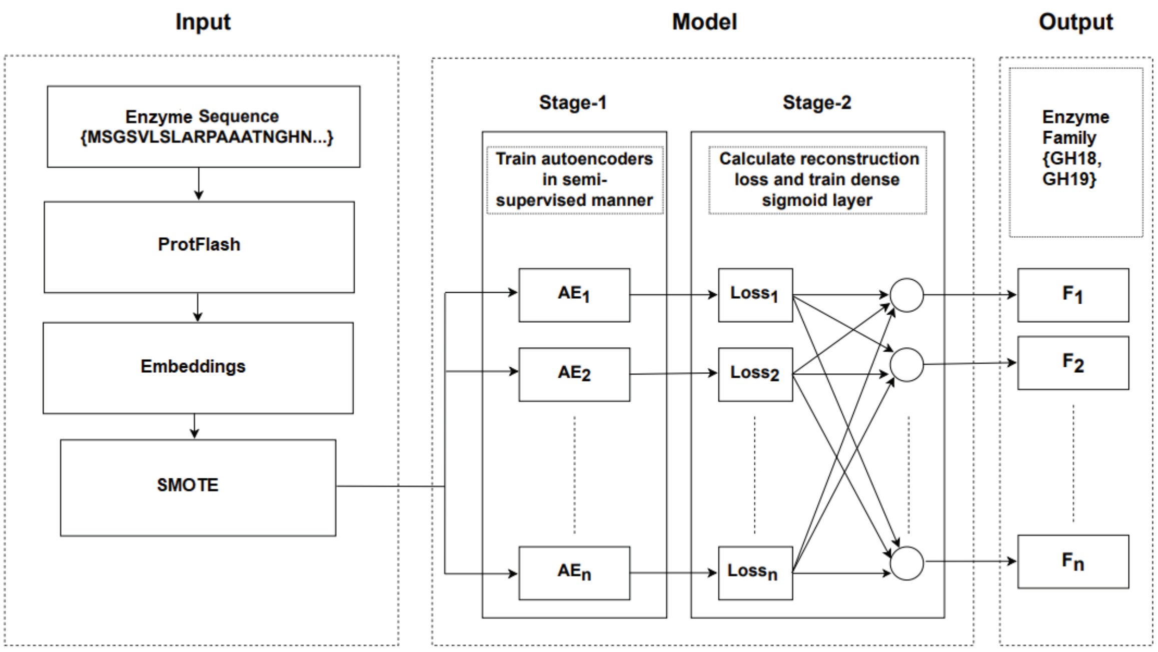
Input
Model
Output
Enzyme Sequence {MSGSVLSLARPAAATNGHN...}
ProtFlash
Embeddings
SMOTE
Stage-1
Stage-2
Train autoencoders in semi-supervised manner
Calculate reconstruction loss and train dense sigmoid layer
AE1
AE2
AEn
Loss1
Loss2
Lossn
Enzyme Family {GH18, GH19}
F1
F2
Fn

Figure 3. Descriptive diagram of the workflow of the second level of the classifier. The processes AE_1, AE_2, ..., AE_n represent the Autoencoders of the corresponding enzyme families F_1, F_2, ..., F_n

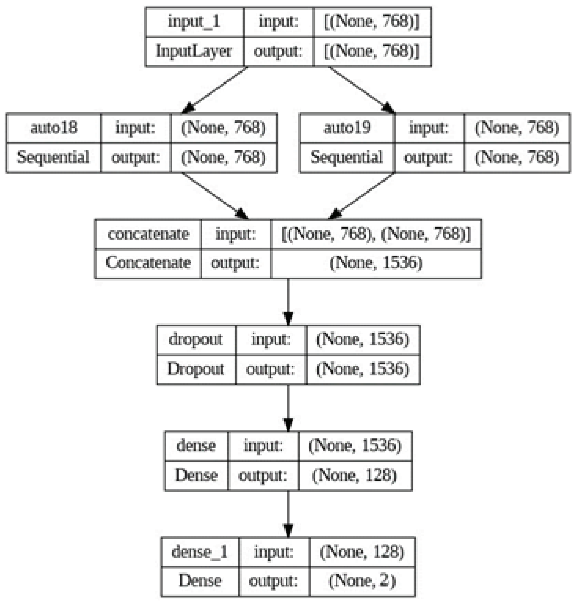
input_1 | input: | [(None, 768)]
InputLayer | output: | [(None, 768)]
auto18 | input: | (None, 768)
Sequential | output: | (None, 768)
auto19 | input: | (None, 768)
Sequential | output: | (None, 768)
concatenate | input: | [(None, 768), (None, 768)]
Concatenate | output: | (None, 1536)
dropout | input: | (None, 1536)
Dropout | output: | (None, 1536)
dense | input: | (None, 1536)
Dense | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 2)

Figure 4. Architecture of the two-level classifier (*the enzyme family classifier*) once implemented using the TensorFlow library

To test the different software, an external test dataset was generated consisting of 20 enzyme sequences from the GH18 and GH19 families, 10 for each one. Additionally, 10 non-enzyme sequences were included in the dataset, which were extracted from the database UniProt[1].

Tables 4, 5, and 6 present the results of the comparison of the different selected software regarding the classification of sequences into enzymes or non-enzymes, evaluated using three key metrics: precision, recall, and F1-Score. The comparison includes EzyPred, ECPred, Proteinfer, and the new classifier using Autoencoders (AE).

Table 3. Comparison of the training of the Second Level

	Loss Function	**Loss Function (Validation)**
Without SMOTE	<u>0.0328</u>	0.1163
SMOTE	0.0518	<u>0.0330</u>
Hyperparameter optimization. (SMOTE)	0.0392	0.0350

Best values are underlined.

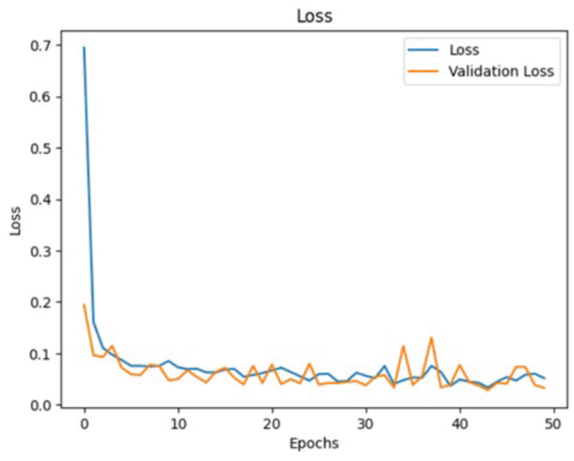

Figure 5. Loss function (categorical cross-entropy) of the second level classifier using SMOTE

The results indicate that the Autoencoder-based classifier achieved remarkable performance in classifying sequences as enzymes or non-enzymes, surpassing or matching other software in terms of all the three metrics.

Table 4. Comparison of different softwares for the classification of sequences into enzymes or non-enzymes (precision)

	EzyPred	ECPred	Proteinfer	AE
Not Enzyme	0.59	0.57	0.47	0.91
Enzyme	1.00	0.82	0.95	1.00

Best values are underlined.

Table 5. Comparison of different softwares for the classification of sequences into enzymes or non-enzymes (recall)

	EzyPred	ECPred	Proteinfer	AE
Not Enzyme	1.00	0.40	0.90	1.00
Enzyme	0.77	0.90	0.67	0.97

Best values are underlined.

Table 6. Comparison of different softwares for the classification of sequences into enzymes or non-enzymes (F1-score)

	EzyPred	ECPred	Proteinfer	AE
Not Enzyme	0.74	0.47	0.62	0.95
Enzyme	0.87	0.86	0.78	0.98

Best values are underlined.

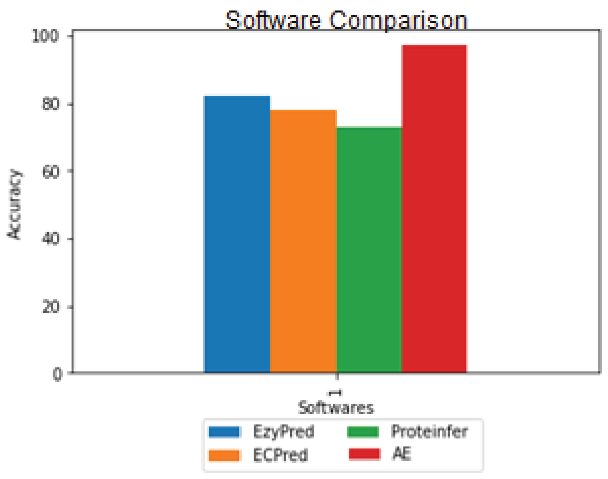

Figure 6. Comparison of Accuracy by different software for enzyme classification or not

Figure 6 presents the analysis of the accuracy obtained by various softwares when evaluating whether a given sequence corresponds to an enzyme or not. The results indicate that the proposed method outperformed existing software, demonstrating its effectiveness in the task of enzyme sequence classification compared to the competition.

In Table 7, the results obtained by the proposed method using Autoencoders for the classification in different enzyme families, specifically GH18 and GH19, are presented. The classifier achieved a remarkable result by achieving an accuracy of 0.90, indicating a high efficacy in the task of classifying enzyme sequences in these families.

Table 7. Results of the family classification

	Precision	Recall	F1-Score
GH18	0.90	0.90	0.90
GH19	0.91	1.00	0.95
No Enzyme	0.89	0.80	0.84
accuracy		0.90	
macro avg	0.90	0.90	0.90
weighted avg	0.90	0.90	0.90

3. Conclusion

The use of Autoencoders in a two-level classifier allows determination of whether a given sequence belongs to the enzyme category or not. This is achieved through an Anomalous Autoencoder, which helps address the lack of representativeness of enzymes. The developed method also provides additional information about the enzyme family to which the sequence belongs. To improve the performance of the model at the two levels of the classifier and reduce deviations from predictions in relation to the actual values, preprocessing techniques are applied, such as increasing the number of training sequences in the imbalanced class through SMOTE.

The proposed approach first transforms the sequences into embeddings with pre-trained models built from heterogeneous sources and then applies Autoencoders for the classification process. In a comparison experiment with some outstanding enzyme classification softwares, this approach shows encouraging external test set accuracy results (90%) in detecting whether a sequence is an enzyme or not. Additionally, as we mentioned before, this approach provides information about the possible GH family to which the sequence could belong, something that those analyzed softwares do not offer.

Notes

[1]https://www.uniprot.org/

AUTHORS

Darian Fernández Gutiérrez* – Center for Informatics Research, Faculty of Mathematics, Physics, and Computer Science, Central University "Marta Abreu" de Las Villas, Cuba, e-mail: dfgutierrez@uclv.cu https://orcid.org/0009-0001-5076-6413.

Ariadna Arbolaez Espinosa – Center for Informatics Research, Faculty of Mathematics, Physics, and Computer Science, Central University "Marta Abreu" de Las Villas, Cuba, e-mail: ararbolaez@uclv.cu https://orcid.org/0009-0005-2752-1986.

Deborah Raquel Galpert Cañizares – Center for Informatics Research, Faculty of Mathematics, Physics, and Computer Science, Central University "Marta Abreu" de Las Villas, Cuba, e-mail: deborah@uclv.edu.cu https://orcid.org/0000-0002-5222-3324.

María Matilde García Lorenzo – Center for Informatics Research, Faculty of Mathematics, Physics, and Computer Science, Central University "Marta Abreu" de Las Villas, Cuba, e-mail: mmgarcia@uclv.edu.cu.

*Corresponding author

References

[1] N. Buton, F. Coste, and Y. Le Cunff, "Predicting Enzymatic Function of Protein Sequences With Attention," *Bioinformatics*, vol. 39, no. 10, Oct. 2023, doi: 10.1093/bioinformatics/btad620.

[2] Y. González Valle, D. Galpert, and R. Molina-Ruiz, "Integración De Rasgos Y Aprendizaje Semi-Supervisado Para La Clasificación Funcional De Enzimas Utilizando K-Means De Spark," *Revista Cubana de Ciencias Informáticas*, vol. 14, no. 4, 2020.

[3] Y. González Valle, D. Galpert, and R. Molina-Ruiz, "Agrupamiento Funcional De Enzimas GH-70 Utilizando Aprendizaje Semi-Supervisado Y Apache Spark," *Revista Cubana de Transformación Digital*, pp. 14–32, 2021.

[4] H. Chehili, S. E. Aliouane, A. Bendahmane, and M. A. Hamidechi, "DeepEnz: Prediction Of Enzyme Classification By Deep Learning," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, 2021, doi: 10.11591/ijeecs.v22.i2.pp1108-1115.

[5] Z. Tao, B. Dong, Z. Teng, and Y. Zhao, "The Classification of Enzymes by Deep Learning," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2992468.

[6] N. Ibtehaz and D. Kihara, "Application of Sequence Embedding in Protein Sequence-Based Predictions," in *Machine Learning in Bioinformatics of Protein Sequences: Algorithms, Databases and Resources for Modern Protein Bioinformatics*, 2022. doi: 10.1142/9789811258589_0002.

[7] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold, "Learned Protein Embeddings For Machine Learning," *Bioinformatics*, vol. 34, no. 15, pp. 2642–2648, Aug. 2018, doi: 10.1093/bioinformatics/bty178.

[8] C. Marquet et al., "Embeddings From Protein Language Models Predict Conservation And Variant Effects," *Hum Genet*, vol. 141, no. 10, 2022, doi: 10.1007/s00439-021-02411-y.

[9] M. M. Moya and D. R. Hush, "Network Constraints And Multi-Objective Optimization For One-Class Classification," *Neural Networks*, vol. 9, no. 3, 1996, doi: 10.1016/0893-6080(95)00120-4.

[10] M. Sakurada and T. Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction," in *ACM International Conference Proceeding Series*, 2014. doi: 10.1145/2689746.2689747.

[11] K. Pawar and V. Attar, "Deep Learning Model Based on Cascaded Autoencoders and One-Class Learning For Detection And Localization Of Anomalies From Surveillance Videos," *IET Biom*, vol. 11, no. 4, 2022, doi: 10.1049/bme2.12064.

[12] L. López, N. Acosta-Mendoza, and A. Gago-Alonso, "Detección De Anomalías Basada En Aprendizaje Profundo," *Revista de Ciencias Informáticas*, vol. 13, no. 3, 2020.

[13] M. V. Nallapareddy and R. Dwivedula, "ABLE: Attention Based Learning For Enzyme Classification," *Comput Biol Chem*, vol. 94, p. 107558, 2021, doi: https://doi.org/10.1016/j.compbiolchem.2021.107558.

[14] R. Atienza, *Advanced Deep Learning with Keras*. 2018.

[15] L. Wang, H. Zhang, W. Xu, Z. Xue, and Y. Wang, "Deciphering The Protein Landscape With Protflash, A Lightweight Language Model," *Cell Rep Phys Sci*, vol. 4, no. 10, p. 101600, 2023, doi: https://doi.org/10.1016/j.xcrp.2023.101600.

[16] K. Cabello-Solorzano, I. Ortigosa de Araujo, M. Peña, L. Correia, and A. J. Tallón-Ballesteros, "The Impact of Data Normalization on the Accuracy of Machine Learning Algorithms: A Comparative Analysis," 2023. doi: 10.1007/978-3-031-42536-3_33.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, 2002, doi: 10.1613/jair.953.

[18] G. Douzas, F. Bacao, and F. Last, "Improving Imbalanced Learning Through A Heuristic Oversampling Method Based On K-Means And SMOTE," *Inf Sci* (N Y), vol. 465, 2018, doi: 10.1016/j.ins.2018.06.056.

[19] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A New Over-Sampling Method In Imbalanced Data Sets Learning," in *Lecture Notes in Computer Science*, 2005. doi: 10.1007/11538059_91.

[20] Aurélien Géaron, *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools, And Techniques to Build Intelligent Systems*. 2022.

[21] D. P. Kingma and J. L. Ba, "Adam: A Method For Stochastic Optimization," in *3rd International Conference on Learning Representations*, ICLR 2015 - Conference Track Proceedings, 2015.

[22] R. Dhanuka, A. Tripathi, and J. P. Singh, "A Semi-Supervised Autoencoder-Based Approach for Protein Function Prediction," *IEEE J Biomed Health Inform*, vol. 26, no. 10, pp. 4957–4965, Oct. 2022, doi: 10.1109/JBHI.2022.3163150.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way To Prevent Neural Networks

From Overfitting," *Journal of Machine Learning Research*, vol. 15, 2014.

[24] T. Dozat, "Incorporating Nesterov Momentum into Adam," *ICLR Workshop*, no. 1, 2016.

[25] H. Bin Shen and K. C. Chou, "EzyPred: A Top–Down Approach For Predicting Enzyme Functional Classes And Subclasses," *Biochem Biophys Res Commun*, vol. 364, no. 1, pp. 53–59, Dec. 2007, doi: 10.1016/J.BBRC.2007.09.098.

[26] A. Dalkiran, A. S. Rifaioglu, M. J. Martin, R. Cetin-Atalay, V. Atalay, and T. Doğan, "ECPred: A Tool For The Prediction Of The Enzymatic Functions Of Protein Sequences Based On The EC Nomenclature," *BMC Bioinformatics*, vol. 19, no. 1, Sep. 2018, doi: 10.1186/s12859-018-2368-y.

[27] T. Sanderson, M. L. Bileschi, D. Belanger, and L. J. Colwell, "ProteInfer, Deep Neural Networks for Protein Functional Inference," *Elife*, vol. 12, 2023, doi: 10.7554/eLife.80942.