Supplemental Materials for DEEPre: sequence-based enzyme EC number prediction by deep learning

Yu Li[1], Sheng Wang[1], Ramzan Umarov[1], Bingqing Xie[2], Ming Fan[3], Lihua Li[3], and Xin Gao[*1]

[1]*King Abdullah University of Science and Technology (KAUST), Computational Bioscience Research Center (CBRC), Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, Thuwal, 23955-6900, Saudi Arabia*

[2]*Illinois Institute of Technology, Computer Science Department, 10 West 35th Street, Chicago, IL 60616, USA*

[3]*Institute of Biomedical Engineering and Instrumentation, Hangzhou Dianzi University, Hangzhou, 310018, China*

[*]All correspondence should be addressed to Xin Gao (xin.gao@kaust.edu.sa)

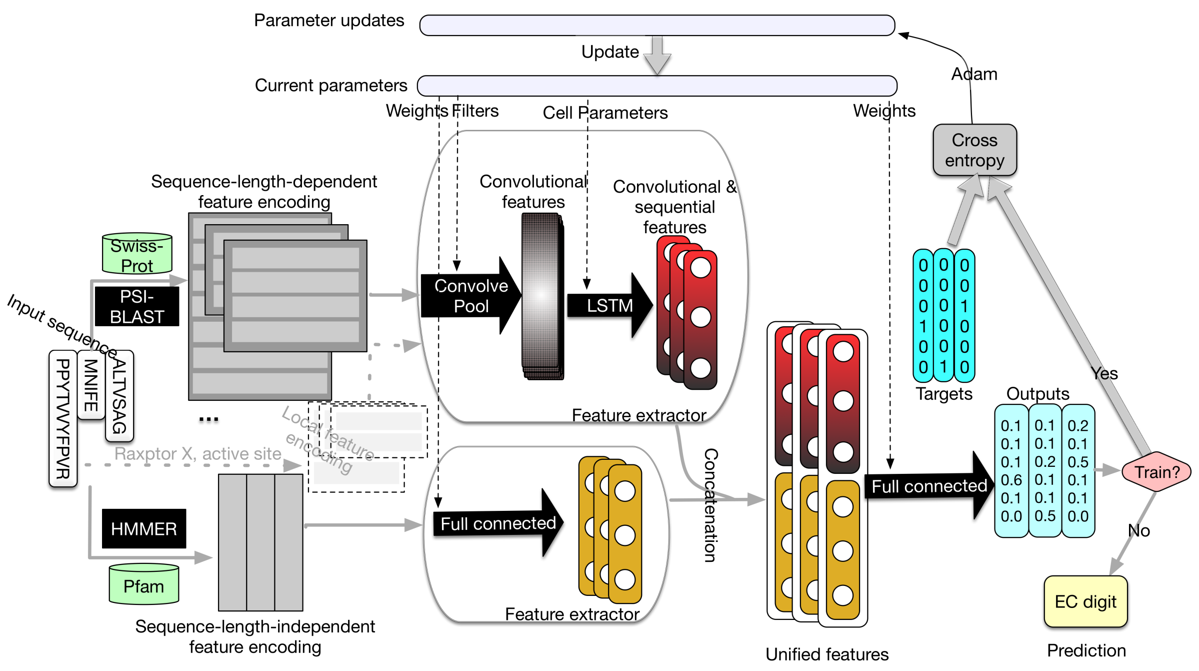

Figure S1: Full overview of our computational model. Here we show the processing of three sequences with different lengths. Notice that after processing, the sequence-length-dependent feature encodings, which have different dimensionality, become convolutional and sequential features with the same length.

S1: Model Structure Details

As shown in Figure S1, we used different feature extractor structures for sequence-length-dependent feature encoding and sequence-length-independent feature encoding. For the former, we combined convolutional neural network and recurrent neural network. In practice, the component contains six convolutional layers and four max-pooling layers, as well as a recurrent layer. Correspondingly, the output of the convolutional layers is a 3D tensor, whose dimensionality is (256, 25, 5), where 256 represents the channel number; 25 is the feature map height; and 5 is the feature map width. The 3D tensor is then fed into the recurrent layer, whose output is a 256D vector. Regarding the latter, we used two fully connected layers to conduct feature extraction, whose output is a 1024D vector. Until now, there is no interaction between any different kinds of features, with the feature extractors acting on each encoding individually. After the feature extraction step, we concatenated different vectors into one vector, whose length is 1536 if we have PSSM, sequence encoding and FuncD as inputs, which is then fed into a fully connected component, that is, the classifier.

S2: Model Parameter Setting

First of all, we have to set a proper feeding batch size. Due to the extremely high dimensionality of our input (for example, the highest dimensionality of PSSM input is 20*5000), we faced serious GPU memory issues when training the model. To make the model work, we set the batch size as 20 for two Pascal Titan X with 24G memory in total. Secondly, the initialization of the model weights would affect the final model performance significantly. We adopted the method mentioned in (He *et al.*, 2015), which suggests a better distribution shape from which the weights should be drawn. Another parameter should be aware of is the weight decay coefficient. If we set it too large, the model tends to be under-fitting while on the other hand, it would be sensitive to noise, which leads to over-fitting if we set it too small. We chose the coefficient as 0.0002 when training the level 0 and level 1 models. For level 2 and level 3 models, since there are less data, which results in higher risk of over-fitting, we increased the weight decay slightly to 0.0005. Furthermore, the training steps should be set appropriately as well. Like the weight decay, the improper value of it would lead to either over-fitting or under-fitting. For level 0 and level 1 models, we set it as 20000, while for level 2 and level 3 models, we set it as 3000.

Hyperparameters that we need to tune are the number of hidden layers and hidden nodes in the fully connected classifier. For models that have larger training dataset, we can increase them to some extent because the dataset makes the model less likely to become over-fitting. However, for level 2 and level 3, whose dataset is much small for each class, we should decrease them accordingly to avoid too high model complexity. Practically, we chose three hidden layers with 1024, 2048, and 1024 hidden nodes for level 0 and level 1 models while only one hidden layers with 1024 nodes for level 2 and level 3 models.

S3: Performance Measure

For the enzyme or non-enzyme prediction, since it is a binary classification problem, we use accuracy, Cohne's Kappa Score (Viera and Garrett, 2005), precision, recall and F_1 score to evaluate the classifiers' performance. For other predictions, since they are multi-class classification problems, we use accuracy, Cohen's Kappa Score, Macro-precision, Macro-recall and Macro-F_1 score to evaluate the classifiers' performance, which are defined below:

$$accuracy(y,\, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(y_i{=}\hat{y}_i),$$

$$Macro\text{-}precision(y,\, \hat{y}) = \frac{1}{|M|} \sum_{l \in M} precision(y_l,\, \hat{y}_l),$$

$$Macro\text{-}recall(y,\, \hat{y}) = \frac{1}{|M|} \sum_{l \in M} recall(y_l,\, \hat{y}_l),$$

$$Macro\text{-}F_1\ score(y,\, \hat{y}) = \frac{1}{|M|} \sum_{l \in M} F_1\ score(y_l,\, \hat{y}_l),$$

where M is the set of labels; y_l is the subset of y with label l; and $\hat{y}_l$ is the predicted label for the subset.

The Cohen's Kappa Score is defined as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

where p_o is the relative observed agreement between true labels and predicted labels, which is identical to accuracy, and p_e is the expected agreement between true labels and predicted labels if we predict the label randomly. Details can be referred to (Viera and Garrett, 2005).

As for the accuracy defined above, it is the weighted average of the accuracy that is commonly used in binary classification, whereas the Macro criteria are the unweighted average of the corresponding criterion used in binary classification. The accuracy is used to reflect the overall performance of the model on the entire dataset, while the Macro criteria are utilized to show the model's performance on small classes. If the model does not perform well on a certain small class, such as class 1.20 which only contains 10 sequences, although it may not be shown in the accuracy since it is biased towards the result of large classes, it would be reflected in the Macro criteria by the unweighted average.

S4: Local Feature Processing

As shown in Figure S1, we also performed analysis on the local features, like solvent accessibility and secondary structure. After obtaining the raw result from DeepCNF (Wang *et al.*, 2016), we incorporated the two pieces of information with active site information from the database and performed analysis in the following way.

1. We collected the active site information from UniProt (UniProt, 2007) (released on February 15, 2017). Since the database is not complete with some entries lacking that information, we sifted the NEW dataset to produce another dataset, containing 8382 enzymes, whose sequences all have active site information.

2. Inputting the selected enzyme sequence into DeepCNF, we obtained solvent accessibility and secondary structure encoding, both of which are L

by 3 matrices.

3. Using active site information, we cropped a smaller matrix around the active site with the window size as seven from the raw output of DeepCNF. If the enzyme has more than one active site, we would concatenate those small matrices into a larger one. If there are overlaps between the small matrices, we would merge the overlaps into one copy. The merged matrix is the output of our model. Because different enzymes have different numbers of active sites, those two feature encodings are still sequence-dependent although we performed some modification.

4. We trained the model with the original three feature and the model with current five features on the sifted dataset and conducted performance comparison.

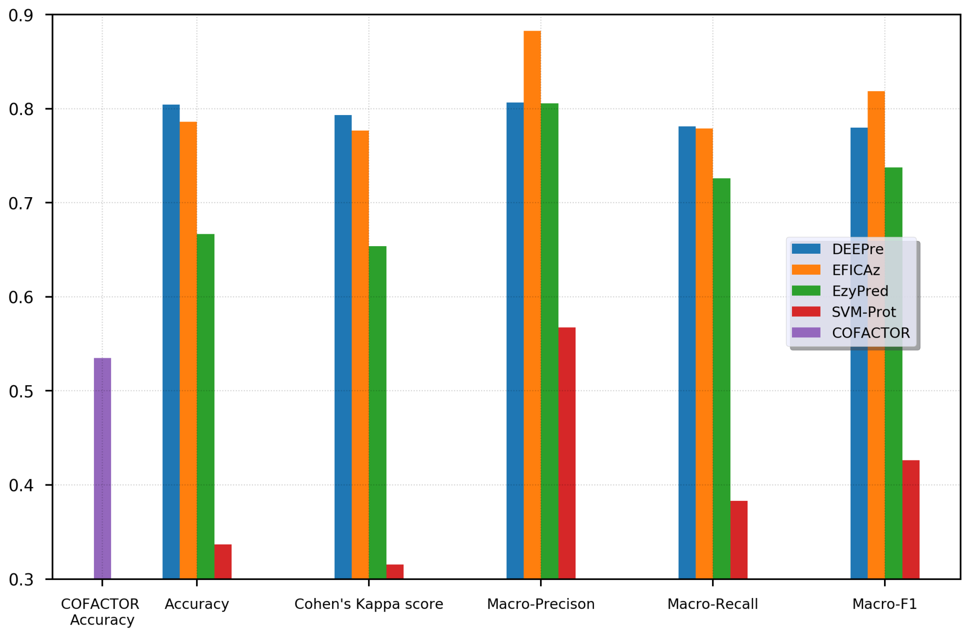

Figure S2: The performance comparison of different servers on the subclass (i.e., the second-digit) prediction of the $COFACTOR$ dataset.

S5: Second-digit Prediction on $COFACTOR$

Figure S2 shows the performance comparison of different servers on predicting the subclass (i.e., the second-digit) on the $COFACtoR$ dataset. DEEPre outperforms EzyPred and SVM-Prot consistently under all the five criteria. It also surpasses EFICAz in accuracy, Kappa score and Macro-recall. The only two criteria that EFICAz outperforms DEEPre are Macro-precision and Macro-F1. There are three possible reasons for this. First, EFICAz is a meta-server which combines six individual methods together. Although it is widely recognized that a meta-server often outperforms individual methods, DEEPre still outperforms EFICAz under all the criteria on the first-digit prediction, as well as under three criteria on the second-digit prediction on the $COFACTOR$ dataset. Second, when preprocessing the $COFACTOR$ dataset, we excluded those sequences that overlap with the DEEPre training data. However, since we did not know the exact training set of EFICAz, we did not perform any overlap reduction between the EFICAz training set and the $COFACTOR$ dataset, which means there are likely overlapping enzymes between the two. This may result in a biased performance of EFICAz. Third, it should be noticed that several methods in EFICAz, such as the pairwise sequence comparison component and multiple PROSITE pattern recognition component, are similarity-based approaches. According to the definition of precision, that is, the ratio of true positives to predicted positives, similarity-based methods are expected to have high precision. especially on problems with small classes, like the enzyme function prediction problem. In summary, we believe the statistical nature of DEEPre and the fact that it does not rely on similarity search makes it a promising method to predict functions of novel enzymes, especially those without close homologs.

S6: Case Study Performance Comparison

Table S1 shows the performance of different servers on the two case studies.

Server	Glutaminase	Aurora kinases B
DEEPre	Completely correct annotation	Completely correct annotation
EFICAz	Completely correct annotation	Failed in predicting the function of isoform 3
EzyPred	Completely correct annotation	Completely correct annotation
SVM-Prot	Failed in predicting the EC number of the "canonical" sequence	Failed in predict the EC number of both isoforms
COFACTOR	Failed in predicting the EC number of the "canonical" sequence; The structure of isoform 2 is not available.	The structures of all isoforms of Aurora kinases B are not available.

Table S1: Comparison of server's performance on case studies

References

He, K. M., Zhang, X. Y., Ren, S. Q., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 Ieee International Conference on Computer Vision (Iccv)*, pages 1026–1034.

UniProt, C. (2007). The universal protein resource (uniprot). *Nucleic Acids Res*, **35**(Database issue), D193–7.

Viera, A. J. and Garrett, J. M. (2005). Understanding interobserver agreement: the kappa statistic. *Fam Med*, **37**(5), 360–3.

Wang, S., Peng, J., Ma, J., and Xu, J. (2016). Protein secondary structure prediction using deep convolutional neural fields. *Sci Rep*, **6**, 18962.