CS230

Deep Learning Applications for Computer-Aided Design of Directed Evolution Libraries

Anthony J. Agbay
Department of Bioengineering
Stanford University
aagbay@stanford.edu

Abstract

In the past decades, the advancement in industrial enzymes, biological therapeutics, and biologic-based "green" technologies have been driven by advances in protein engineering. Specifically, the development of directed evolution techniques have enabled scientists to generate genetic diversity to iteratively test randomized libraries of variants to generate optimized enzymes for specific functions. However, throughput of directed evolution is limited due to the potential number number of variants in a randomized library. Training neural network models that utilize the growing number of datasets from deep mutational scanning experiments that characterize entire mutational landscapes and new computational representations of protein properties provides a unique opportunity to significantly improve the efficiency and throughput of directed evolution , alleviating a major bottleneck and cost in many research and development projects.

1 Introduction

From reducing manufacturing waste and developing green energy alternatives, to biocatalytic cascades, designing new therapeutics, and developing inducible protein switches, engineering proteins enable enormous opportunities across a diverse array of fields. (Huffman et al., 2019; Nimrod et al., 2018; Langan et al., 2019; Smith et al., 2012) Directed evolution, or utilizing evolutionary principles to guide iterative mutations to a protein, has been an essential method for driving many protein engineering projects. However, the search space accessible through directed evolution is limited by physical constraints of screening methods, the iterative buildup of mutations, and other factors. (Xiao et al., 2015) The rise of methods, such as deep mutational scanning (DMS), is enabling the application of new computational approaches to directed evolution by addressing long-standing deficiencies in standardized and labeled data sets quantifying the effects of mutations. (Fowler and Fields, 2014) By using these labeled DMS datasets augmented with protein biochemical properties and microenvironment descriptions, we can train a three-class neural network classifier to predict if a potential mutation is deleterious, neutral, or beneficial. These predictions can then be used to guide the development of effective directed evolution libraries.

2 Related work

There have been several different computational approaches for utilizing DMS data to predict mutation effect sizes. Gray et al. (whose paper this dataset is sourced from) utilized stochastic gradient descent and a random forest algorithm, termed Envision(Gray et al., 2018). Four other algorithms are standard for this type of application. PolyPhen-2 uses a Naive Bayes classifier using various biophysical

CS230: Deep Learning, Spring 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

properties to determine the effect category of a mutation. (Adzhubei et al., 2010) SIFT is also a classification algorithm, but uses sequence-alignment scores to rank and classify mutations. (Ng and Henikoff, 2001) EVmutation uses a probabilistic model based on an energy function to predict the favorability of mutations. (Hopf et al., 2017) SNAP2, the closest approach to this project, is a binary neural network classifier using a set of biochemical properties and sequence alignment. All of these approaches attempt to capture a specific subset of protein characteristics, but have been limited by the amount of data and the need to hand engineer more complex features.

3 Dataset and Features

The main dataset is a curated and standardized collection of DMS data used for training and evaluating the Envision algorithm. (Gray et al., 2018) he authors filtered the data based on a series of selection criteria described in the original paper and have standardized the effect size labels. In total, there are 65,420 variants included in this dataset with 32 features. However, 6,522 of these variants have predicted effect sizes due to gaps in the original DMS experiments. These predicted effect sizes were generated using a similar model to the Envision algorithm but applied to each protein. As such, these predicted variants will be excluded, leaving 58,898 variants split amongst 8 different proteins. These data were further processed by filling mislabeled data, missing features, and generating one-hot encodings for categorical features. A full description of the features can be found in the supplemental information from the original paper.

This DMS dataset was further augmented with a description of microenvironments around each mutation. These microenvironments were generated using the FEATURE algorithm in conjunction with DSSP. (Halperin et al., 2008; Touw et al., 2015; Kabsch and Sander, 1983) These algorithms generate a vectorized representation of the microenvironments in a protein by generating counts of different biochemical properties, such as presence of secondary structure elements, specific residues, and charges, in a series of concentric spheres around a single residue. A full description of the merged features can be found in the original FEATURE paper. (Halperin et al., 2008) The algorithm is built to determine these microenvironments at the atomic level. To merge with the original datset, FEATURE vectors were combined at the residue level prior to merging.

Protein	Count
TEM-1	25140
Kka2	20475
Uba1	1364
PSD95pdz3	1576
Pab1	856
hsp90	170

Table 1: Merged Dataset Counts

These algorithms were ran using the default parameters (6 shells, 1.25 Å widths), but five outer shells were omitted due to overfitting issues. To generate these data, we used Uniprot and the Protein Data Bank to retrieve the wild-type crystal structures for each protein of interest. (Berman et al., 2000; noa, 2019) All mutations are assumed to not result in major structural changes to the protein. Because the crystal structures did not contain the all residues mutated in the original dataset, the dataset was further filtered during this merging process (See Table 1). After merging, the final dataset contained 107 features, the majority of which come from the FEATURE encodings.

Overall, there are two limitations with the dataset. First, the distribution amongst each protein is not balanced (See Table 1). This is unavoidable due to difficulties in obtaining crystal structures and generating DMS data through an entire protein. The imbalance will cause variability in the training and evaluation of the final algorithm, as the process is based on a leave-one-protein-out (LOPO) approach (further described in the methods section). Second, there is a further imbalance in the number of examples spread across the three labels. (See Figure 1). Similar to the problem described above, this is unavoidable and is the expected. This created a significant challenge to train an algorithm with strong performance (especially when trying to label "beneficial" mutations). Attempts to address the class imbalances did not result in improved performence (See Methods section).

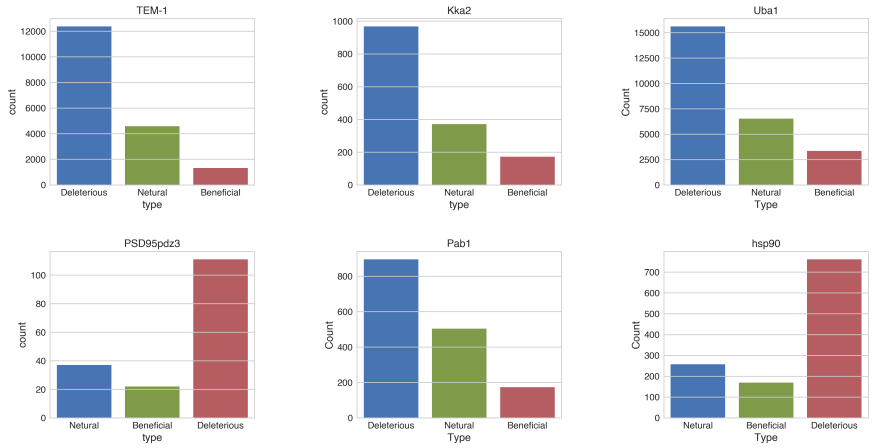

Figure 1: Distribution of Labels per Protein (Prior to Upsampling)

4 Methods

All code used to process data and train/evaluate algorithms can be found at `https://github.com/anthony-agbay/cs230-project`.

4.1 Data Processing

Because the original dataset contained continuous scaled effect labels, the data was processed to generate three class labels: "Deleterious," "Beneficial," and "Neutral." The scaled effect label was normalized to 1 in the original dataset, so we defined a hyperparamter, *threshold offset* ( ), to determine the range for each class:

$$\begin{aligned} \text{Deleterious:} \quad & < 1 - \\ \text{Beneficial:} \quad & > 1 + \\ \text{Neutral:} \quad 1 - \quad & < X < 1 + \end{aligned}$$

After the initial processing described in the above, an additional processing step to address the class imbalance was attempted. Because deleterious samples outnumbered both the neutral and beneficial samples, the neutral and beneficial classes were upsampled to match the number of deleterious samples for each protein. In order to do this, we chose $= 0.2$ to capture an intermediate range of netural mutations. This resulted in a final dataset consisting of almost 69,000 samples. Input features were further standardized using the StandardScaler pipeline from scikit-learn prior to training any models.

4.2 Baselines

4.2.1 Logistic Regression

Protein	Accuracy	Precision
TEM-1	.521	.386
Kka2	.492	.357

Table 2: Logistic Regression Models

First, a logistic regression baseline using the standard implementation from ‘scikit-learn‘ was developed.

To train and evaluate this model, we implemented the LOPO data split method mentioned in the previous section. In a real world application, this algorithm should be able to perform well on proteins that currently do not have labeled DMS data accessible. As a result, the traditional train/dev split would not reflect the expected

performance (as described in Milestone 1). In the LOPO model, samples from a single protein are witheld as the final dev/test set and all remaining samples are used to train the model. Because of this non-standard data split and the imbalanced distribution of samples per protein, there will be variability in the performance based on which protein was selected to be used for the dev/test set.

4.2.2 Two-Layer Neural Network

We also trained a two-layer neural network as an additional baseline to our final model. This model consisting of a single, fully connected hidden layer with 256 nodes and three-node softmax output layer using the 'scikit-learn' wrapper for the Keras library. This baseline network was trained for 20 epochs with a batch size of 64 (based on monitoring the loss over time). Accuracy and precision were determined for TEM-1 and Kka2 only because of continued imbalance problems for the other proteins despite upsampling. (See Table 3.

Protein	Accuracy	Precision	Recall
TEM-1	.413	.404	.413
Kka2	.435	.442	.435

Table 3: Two Layer NN Model

4.3 Neural Network Model

To implement the final neural network model, we utilized the Tensorflow and Keras libraries. This neural network consists of L number of fully connected hidden layers with a ReLU activation function, each with N number of nodes, where L and N are the two additional hyperparameters. These layers are subject to $l2$ normalization followed by a dropout layer with a frequency of 0.2. The final layer is a 3-node softmax output layer. Despite data scaling, normalization for each node, and dropout, overfitting continued to be an issue due to the small number of samples. To address this, we implemented an early stopping criteria based on the validation loss. The batch size was chosen to be 64 based on qualitative analysis of the baseline results.

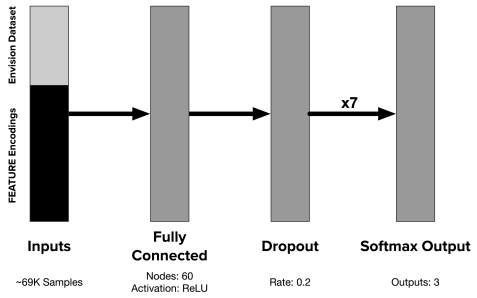

Figure 2: Neural Network Architecture

4.4 Hyperparamter Tuning

To train the hyperparameters, we chose to use a modified grid search over a small number of hidden layers and nodes. We were restricted to a smaller number of hidden layer and nodes due to the lack of data, despite upsampling. This resulted in the following hyperparameter search grid:

Parameter	Range
Number of Hidden Layers (L)	3, 5, 7, 9, 11
Number of Nodes per Layer (N)	10, 20, 30, 40, 50, 60, 70, 80, 90, 100

Table 4: Two Layer NN Model Hyperparameter Search

Unlike a normal train/test split, we utilized the LOPO split method. For hyperparameter tuning, we decided to choose Kka2 as our withheld set. After evaluating each hyperparameter combination for accuracy, precision, and recall, the final model architecture consists of 7 hidden layers with 60 nodes each (See Figure 2). Once the neural network reached 9 hidden layers or larger numbers of nodes, the learning process began to break down and resulted in networks that only predicted a single class for all examples.

5 Experiments/Results/Discussion

The three metrics we use to evaluate our three-class classifer were categorical accuracy, precision, and recall. For the final model, we witheld samples from TEM-1, trained the model using the hyperpameters described above. After training and evaluating the model the evaluation metrics showed an overall improvement over the two-layer neural network baseline, except for decreased recall (See Table 5). Compared to the logistic regression baseline, the accuracy decreased, but the precision increased. Furthermore, the model continued to overfit to the training data, suggesting that additional training data will be needed. These data should consist of additional proteins outside of the protein families already contained within this dataset.

Evaluation Metric	Measure
Accuracy	.439
Precision	.491
Recall	0.310

Table 5: Final Model Evaluation (TEM-1)

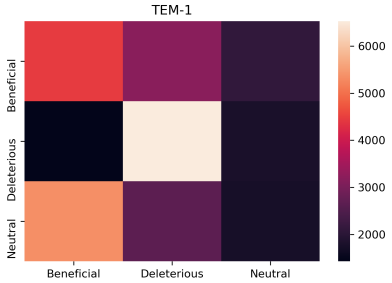

Figure 3: TEM-1 Final Model Heat Map

Analysis of the confusion matrix also supports improved predictive power (See Figure 3) Deleterious mutations were well predicted. Additionally, the final model. These shortcoming may be due to a lack of generalizability given the lack of unique beneficial and neutral mutations in the starting data set. Despite upsampling these classes, the algorithm may not have enough data to properly generalize to a new protein. The lack of training data also reduced the effectiveness of the FEATURE encodings because it limited the number of shells that could be used due to problems of overfitting.

Furthermore, the final neural network model failed to achieve similar performance to the Envision algorithm, which was trained on the unaugmented starting dataset. In addition to the lack of data mentioned above, additional factors may have contribued to the weaker performance. First, some of the assumptions used to aggregate the FEATURE encodings may not hold in reality. In particular, the assumption that mutations do not cause changes to the overall structure may not hold in all cases and contribute to the lack of generalization across proteins. One potential fix would be to add an additional processing step to the input pipeline: generate a structural model for each mutation using Rosetta to better capture the true microenviroment. ((Simons et al., 1999))

6 Conclusions and Future Work

While the final model trained in this project did not achieve strong performance compared to other computational approaches, it does illustrate the potential that neural networks have in advancing directed evolution and protein engineering projects. Like all other deep learning applications, the size of the training dataset is the most significant obstacle to creating a well-performing model. However, obtaining well-curated datasets for this particular application is a major obstacle due to study-to-study variance and a lack of a standardized, central repository. However, DMS experiments continue to improve and as more data becomes availaible, neural networks will be able to make significant contributions to directed evolution and protein engineering projects.

7 Contributions

All work on this project was conducted by Anthony Agbay. The author would like to thank Shubhang Desai and the CS230 teaching staff for their guidance on project scope and implementation details.

References

(2019). UniProt: A Worldwide Hub of Protein Knowledge. *Nucleic Acids Research*, 47(D1):D506–D515. Publisher: Oxford Academic.

Adzhubei, I. A., Schmidt, S., Peshkin, L., Ramensky, V. E., Gerasimova, A., Bork, P., Kondrashov, A. S., and Sunyaev, S. R. (2010). A Method and Server for Predicting Damaging Missense Mutations. *Nature Methods*, 7(4):248–249.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242. Publisher: Oxford Academic.

Fowler, D. M. and Fields, S. (2014). Deep Mutational Scanning: A New Style of Protein Science. *Nature Methods*, 11(8):801–807.

Gray, V. E., Hause, R. J., Luebeck, J., Shendure, J., and Fowler, D. M. (2018). Quantitative Missense Variant Effect Prediction Using Large-Scale Mutagenesis Data. *Cell Systems*, 6(1):116–124.e3.

Halperin, I., Glazer, D. S., Wu, S., and Altman, R. B. (2008). The FEATURE Framework for Protein Function Annotation: Modeling New Functions, Improving Performance, and Extending to Novel Applications. *BMC Genomics*, 9(2):S2.

Hopf, T. A., Ingraham, J. B., Poelwijk, F. J., Schärfe, C. P. I., Springer, M., Sander, C., and Marks, D. S. (2017). Mutation Effects Predicted From Sequence Co-Variation. *Nature Biotechnology*, 35(2):128–135.

Huffman, M. A., Fryszkowska, A., Alvizo, O., Borra-Garske, M., Campos, K. R., Canada, K. A., Devine, P. N., Duan, D., Forstater, J. H., Grosser, S. T., Halsey, H. M., Hughes, G. J., Jo, J., Joyce, L. A., Kolev, J. N., Liang, J., Maloney, K. M., Mann, B. F., Marshall, N. M., McLaughlin, M., Moore, J. C., Murphy, G. S., Nawrat, C. C., Nazor, J., Novick, S., Patel, N. R., Rodriguez-Granillo, A., Robaire, S. A., Sherer, E. C., Truppo, M. D., Whittaker, A. M., Verma, D., Xiao, L., Xu, Y., and Yang, H. (2019). Design of an in Vitro Biocatalytic Cascade for the Manufacture of Islatravir. *Science (New York, N.Y.)*, 366(6470):1255–1259.

Kabsch, W. and Sander, C. (1983). Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers*, 22(12):2577–2637.

Langan, R. A., Boyken, S. E., Ng, A. H., Samson, J. A., Dods, G., Westbrook, A. M., Nguyen, T. H., Lajoie, M. J., Chen, Z., Berger, S., Mulligan, V. K., Dueber, J. E., Novak, W. R. P., El-Samad, H., and Baker, D. (2019). De Novo Design of Bioactive Protein Switches. *Nature*, 572(7768):205–210.

Ng, P. C. and Henikoff, S. (2001). Predicting Deleterious Amino Acid Substitutions. *Genome Research*, 11(5):863–874.

Nimrod, G., Fischman, S., Austin, M., Herman, A., Keyes, F., Leiderman, O., Hargreaves, D., Strajbl, M., Breed, J., Klompus, S., Minton, K., Spooner, J., Buchanan, A., Vaughan, T. J., and Ofran, Y. (2018). Computational Design of Epitope-Specific Functional Antibodies. *Cell Reports*, 25(8):2121–2131.e5.

Simons, K. T., Bonneau, R., Ruczinski, I., and Baker, D. (1999). Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins*, Suppl 3:171–176.

Smith, P. R., Bingham, A. S., and Swartz, J. R. (2012). Generation of Hydrogen From NADPH Using an [FeFe] Hydrogenase. *International Journal of Hydrogen Energy*, 37(3):2977–2983.

Touw, W. G., Baakman, C., Black, J., te Beek, T. A. H., Krieger, E., Joosten, R. P., and Vriend, G. (2015). A Series of PDB-Related Databanks for Everyday Needs. *Nucleic Acids Research*, 43(Database issue):D364–368.

Xiao, H., Bao, Z., and Zhao, H. (2015). High Throughput Screening and Selection Methods for Directed Enzyme Evolution. *Industrial & Engineering Chemistry Research*, 54(16):4011–4020.