EPiC Series in Computing
Volume 92, 2023, Pages 47–57
Proceedings of International Conference on Bioinformatics and Computational Biology (BICOB-2023)

Classifying Protein Families with Learned Compressed Representations

Ramin Dehghanpoor, Fatemeh Afrasiabi, Charles Fogel, Tung Dao, Suman Gautam, Aanab Nehela, Ahmad Nehela, Daniel Haehn, and Nurit Haspel

University of Massachusetts, Boston, Massachusetts, USA
Ramin.Dehghanpoor001@umb.edu, Fatemeh.Afrasiabi001@umb.edu, Charles.Fogel001@umb.edu, Tung.Dao001@umb.edu, Suman.Gautam001@umb.edu, Aanab.Nehela001@umb.edu, Ahmad.Nehela001@umb.edu, Daniel.Haehn@umb.edu, Nurit.Haspel@umb.edu

Abstract

Classifying proteins into families is an important task when studying newly discovered proteins. If we can identify the family a protein belongs to, we can predict features without knowing the exact structure of such a protein. However, this grouping process is challenging. We propose a two-stage algorithm that classifies proteins into families by combining a dimensionality reduction technique using a variational autoencoder with learned fingerprint representations using a Convolutional Neural Network (CNN). Our models use fewer parameters than existing methods but perform better, with our variational autoencoder achieving 94% accuracy in reconstructing the most common amino acid in a sequence alignment, and the neural network provides 98-100% accuracy in classifying protein families. We developed a software framework to access our algorithms. All code and data are publicly available at https://github.com/ramindehghanpoor/CLI.

1 Introduction

Protein family classification for newly discovered protein sequences is a crucial step in bioinformatics analysis. For example, an important application is finding the structure and functionality of a new protein sequence [1, 2]. Protein structure is tightly linked to its function and determines how it interacts with other proteins. If we know the structures of proteins, we can have higher levels of understanding of how they function. Consequently, we can learn how to modify or control them. Predicting the structure of proteins is still an open, NP-hard problem in structural biology that is still challenging even for state-of-the-art software and hardware. This is due to the enormous size of the search space of protein structures and the fact that their structural landscape is rugged [3]. Generally, experimentally detecting a protein structure is challenging. The number of known protein sequences grows fast, but the number of known structures lags behind [4]. Therefore, we need to look for other ways to model protein structures. It is almost impossible to find crystal structures of all the protein sequences experimentally, and even the best prediction models like AlphaFold are far from perfect [5]. But we may have the structures of different proteins in the same family [2, 1]. This task is especially challenging when the sequence shares limited sequence similarity to any of the sequences in databases [6]. If we have the protein structure, authors sometimes use the structural information to identify its family [7], but as mentioned above, the structure is not always available. Hence, improving the accuracy of methods that classify sequences into existing families is an area of active research [1, 2, 8, 9]. A protein family consists of proteins with the exact evolutionary origin. These proteins have similar functionalities or similarities in sequence or structures [10]. The fact that proteins in the same family have similar sequences and structures can help us predict the structures of proteins for which their 3D structure has not been found experimentally yet.

Dimensionality Reduction

Working with a lower-dimensional dataset saves training time, computational resources, and space and tackles the problem of over-fitting. We can divide dimensionality reduction methods into the ones that work best for linear data and the ones that work well for nonlinear data. One of the most famous linear dimensionality reduction techniques is Principal Component Analysis (PCA) [11]. PCA builds new independent features as a linear combination of old features. PCA tries to find the best linear subspace of the data that minimizes the error of estimating the data by their projections on this subspace. Nonlinear methods can work significantly better with complex nonlinear data, like protein sequences, than linear ones. Recently, there has been a surge in the utilization of Autoencoders to reduce the dimensionality of nonlinear data.

Auto Encoders (AE): Autoencoders are unsupervised feed-forward artificial neural networks with two primary usages; compressing the data and regenerating new similar data from the compressed one. Autoencoder networks have an odd number of layers that comprise three key components: the encoder, the latent space (i.e., bottleneck), and the decoder [12]. These networks have a particular architecture that forces the data to be encoded to a smaller dimension in the latent space and reconstruct a representation as close to the original input as possible from the reduced data. Therefore, the first half of the network, i.e., the encoder, is a model which maps the data from high to low-dimensional space. We train the network to minimize a loss function, for example, the mean squared error, between the input and the output of the network and to ignore the noise and keep the essential parts of the data. Autoencoders have many applications in bioinformatics and specially protein studies like prediction of protein-protein interaction [13, 14, 15], protein secondary structure prediction [16], protein function prediction [17], protein dynamics studies [18] and many more.

Variational Auto Encoders (VAE): The loss function in a regular autoencoder is not concerned with how the latent space is organized. Its only purpose is to encode and decode with the minimum loss possible. Without regularizing the network, it's more likely to have an overfitting problem. A Variational Auto Encoder (VAE) [19], on the other hand, is a type of autoencoder that incorporates the regularization factor in the training process to prevent overfitting [20]. VAE learns a latent variable model, i.e., a *distribution* over the latent space, rather than a single point. So instead of letting the network learn a random function, we force it to learn the parameters of a probability distribution that models our data. First, the network encodes the input as a normal distribution over the latent space and then samples a point from this distribution. Next, the second part of the network decodes the sampled point, the loss function calculates the reconstruction error, and the network backpropagates the reconstruction error through the network. Using a distribution with some variance adds a regularization to the latent space. The loss function of VAE is a combination of two loss

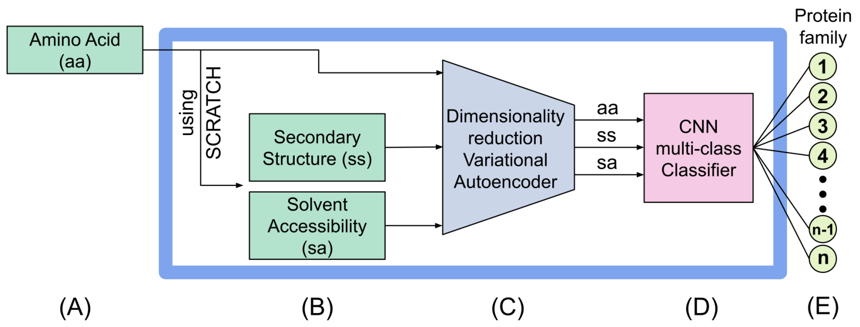
Protein family
Amino Acid (aa)
using SCRATCH
Secondary Structure (ss)
Solvent Accessibility (sa)
Dimensionality reduction Variational Autoencoder
aa
ss
sa
CNN multi-class Classifier
1
2
3
4
n-1
n
(A)
(B)
(C)
(D)
(E)

Figure 1: An overview of the project. (A) amino acid sequence as the input data, (B) two other features computed for secondary structure and solvent accessibility, (C) phase 1: dimensionality reduction using a variational autoencoder, (D) phase 2: protein family classification using a 2D convolutional neural network, and (E) protein family as the output data.

functions: a reconstruction loss, forcing the reconstructed data to be as close as possible to the input, and a regularization loss (Kullback-Leibler divergence), forcing the distributions returned by the encoder to be as close as possible to a standard normal distribution. VAEs are described in detail in [19]. Some applications of VAE in protein studies include protein structure prediction [21], generating protein variants and tertiary structures [22, 23], protein conformational space exploration [24, 25], protein fold design [26], and many others.

This Contribution: We use three protein features to classify proteins into different families. These features include the amino acid sequence, the predicted secondary structure, and the predicted solvent accessibility of each amino acid. This work has two main phases. In the first phase, we reduce the dimensionality of our high-dimensional features using a variational autoencoder to create a compact representation of a protein family. In the second phase, we use the dimensionality-reduced features to classify the family of the protein sequences using a CNN multiclass classifier. A schema of our project phases is shown in Figure 1. We also developed an open-source library to search and compare the compressed representation of different protein sequences easily and efficiently (section 2.3).

2 Materials and Methods

2.1 Dataset

This section provides an overview of the dataset used in our study, a set of reviewed protein sequences from the Uniprot database. We used the same family names as the ones used by Zhang et al. [27]. 53 protein families have been fetched from the Uniprot database, with a total number of 49,616 sequences with different lengths. We then removed any sequence with fewer than 50 or greater than 1,200 amino acids. Any protein family with fewer than 500 sequences was removed. We then used ClustalW [28] to align the sequences in each protein family. We performed two steps of preprocessing on the multiple sequence alignments. First, all the columns with more than 99.5% gap were identified, and sequences causing those gaps were removed (any sequence with non-gap characters in those columns). Second, the columns with more than 75% gap were removed. This technique is often used to ensure consistency of good coverage across the alignment [29, 30]. After these preprocessing steps, our dataset resulted in 47,234 sequences in 48 families with 522 to 3,475 sequences in each family. It is impossible to use all the sequences of each family because of the bias in classifying families with more sequences available. It is essential to have a balanced dataset for classification tasks [31, 32]. Therefore, 500 sequences were randomly selected from each family to have families of the same size. Our dataset is a high-dimensional dataset since each sequence has a length of 143 to 1,092 residues (with gaps added).

We use three features of each protein sequence per family. The first feature is the amino acid sequence with 21 possibilities for each position (20 amino acids plus a gap). The secondary structures and solvent accessibility features are predicted using the SCRATCH software [33]. We will evaluate and compare additional software tools to determine their efficacy and effectiveness as part of our future work. The secondary structure feature can be either a *gap*, *helix*, *strand*, or *other*. The solvent accessibility feature has three possibilities: *gap*, *exposed*, and *buried*. Since no ordinal relationships between the possibilities of each feature exist, we use the one-hot encoding technique to convert these categorical features to binary variables.

So, for each input protein sequence with length L, we generate three matrices with the following sizes to be used as the input for the variational autoencoder:

- $L \times 21$ for amino acid sequence
- $L \times 4$ for secondary structure sequence
- $L \times 3$ for solvent accessibility sequence

We set the ratio of the train, validation, and test sets in each dataset as 80%, 10%, and 10%, respectively.

2.2 Phase 1: Dimensionality reduction

The performance of machine learning algorithms can decrease with a very high-dimensional dataset. Dimensionality reduction is a step in data preprocessing, which reduces the number of features and passes only the important ones while trying to keep as much variation in the initial dataset as possible. First, the network encodes the input to two latent spaces z_mean (mean vector) and z_log_sigma (standard deviation vector). Then we randomly take a sample from the latent normal distribution via

$$z = z\_mean + (exp(z\_log\_sigma) * \epsilon)$$

where ϵ is a random normal tensor with $mean = 0$ and $SD = 0.1$. Our final latent space is z, which is an array of 30 float numbers. We call it a *fingerprint* for a protein family. Our VAE has seven layers: input, first intermediate, z_mean, z_log_sigma, z, second intermediate, and output. The encoder is the first part of the network, which has five layers from input to latent, and the decoder is the second part of the network, with three layers from latent to output. All the layers are 1-dimensional vectors of float numbers. The input and output size is equal to the length of the protein sequence times the number of binary variables in the one-hot encoded feature for which we reduce the dimensionality. For instance, when we want to use our variational autoencoder on the amino acid sequence feature, the input size equals sequence length times 21. The size of the two intermediate layers is 128, and the size of z_mean, z_log_sigma, and *latent* is 30. A schematic shape of our variational autoencoder is shown in Figure 2. We use the *ReLu* activation function for intermediate layers, a *linear* activation function for z_mean and z_log_sigma, and the *sigmoid* activation function for the output layer.

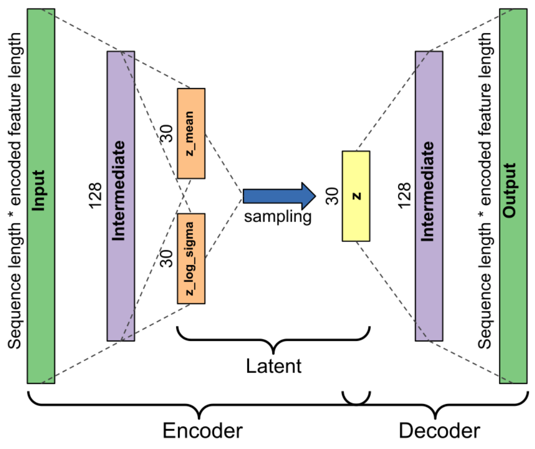

Figure 2: Illustration of our variational autoencoder. The input layer contains the sequence and other information, i.e. secondary structure and solvent accessibility. The encoding results in a latent space of 30 floating point numbers that contain the reduced representation of the proteins. The decoder reconstructs the data as closely as possible to the input data.

2.3 Phase 2: protein family classification

In this phase, we classify the family of a given protein sequence using the dimensionality-reduced data obtained from Phase 1 (Subsection 2.2). This task is characterized as a multiclass classification problem, in which several protein families are considered as potential classes. Given a protein sequence as input, the aim is to accurately assign it to one of these families. Convolutional Neural Networks (CNNs) are artificial neural networks, vastly used for various classification tasks. They are often used to analyze and classify images [34, 35]. A sample image can be a colorful picture with three channels of red, green, and blue, a gray-scale handwritten digit with one channel, or any other matrix of numbers. Convolution means applying a filter to an input. A CNN is made of multiple layers of neurons. A neuron calculates an activation function on a weighted sum of its inputs plus a bias. Applying the same filter to different input sections produces the feature map, revealing the detected features in the input. The network iteratively learns the values of the filters in the training phase to find the best results. The trained network can be used to make the family prediction of unseen data.

Properties of our CNN: Our CNN gets a matrix of three dimensionality-reduced features, i.e., amino acid sequence, secondary structure, and solvent accessibility (subsection 2.1), and has two 2D convolution blocks. Each block has a number of Conv2D filters with size = (7x7), padding = ”same”, strides = 1, followed by batch normalization, dropout with rate = 0.3, and *ReLu* activation. The first and second blocks have 20 and 120 filters, respectively. The output of the second block flattens to a 1D array. The flattened array is fully connected to a dense layer of a size equal to the number of protein families in our dataset, e.g., 500 for our second dataset (details are in subsection 2.1), with the *sigmoid* activation function. Each output layer neuron computes the probability that the target protein sequence belongs to the corresponding protein family. We assign the protein family class with the highest probability to the protein sequence. An overview of our CNN is shown in Figure 3.

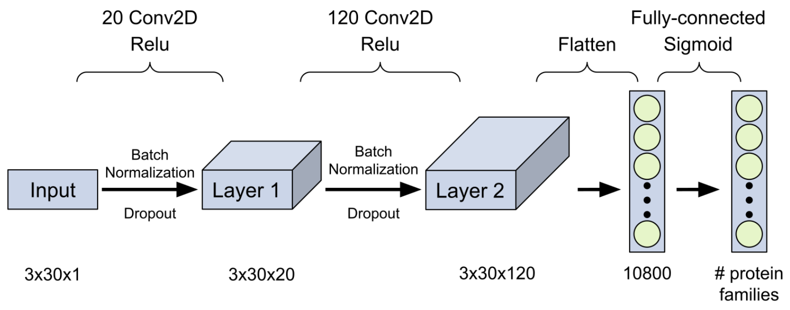

Figure 3: Illustration of our CNN. The input is a representation of a protein and the output is the probability of the protein being classified into any of the families. Input, i.e. the dimensionality reduced data, comes from the previous phase.

Software Framework: We developed a Python-based open-source library that allows us to compare and search for protein families using our algorithms. This library can be installed using a PyPI package with the following command: `pip install compbiolab-CLI`. After installation, users can find the protein family for a new sequence by comparing it with existing sequences in the database. Our framework also supports measuring the distance of partial sequences using a variety of different metrics such as the weighted and unweighted Minkowski measures.

3 Results

3.1 Phase 1 – Variational Autoencoder

We use the amino acid sequences to measure the performance of our variational autoencoder. Our goal is to determine whether the decoder can reconstruct the original input sequence alignment. After training on the train set, we compare the network's input and output for data points in the test set. Our experiments indicate that the network can reconstruct 89% of the input amino acids correctly. A second test compares the most common amino acid in every column of the input sequence alignment with the output reconstructed alignment of the variational autoencoder. For each sequence in a family, we use the variational autoencoder and then the decoder to reconstruct all the sequences. We see that in 95% of the positions of the dataset, the most common amino acid is the same in the original and reconstructed alignment.

As we see next, even in cases where the decoder did not reconstruct the same amino acid, it could find a similar amino acid. We used BLOSUM62 scores to compare the reconstructed amino acids as the output with the original sequences [36]. BLOSUM62 is a scoring matrix that models the substitution score for any pair of amino acids. This number shows the evolutionary distance relationship between any pair of amino acids, and it gives us a similarity score for them [37, 38]. Figure 4 (A) shows the average BLOSUM62 score for substituting the original amino acid for the reconstructed one for each position in one of the protein families in our dataset (Enolase). Each bar represents one position in the protein sequence. Colors are based on BLOSUM62 scores, with green for higher and red for lower values. Figure 4 (B) is similar to Figure 4 (A), but it compares the most common amino acid in the input and output of the variational autoencoder network.

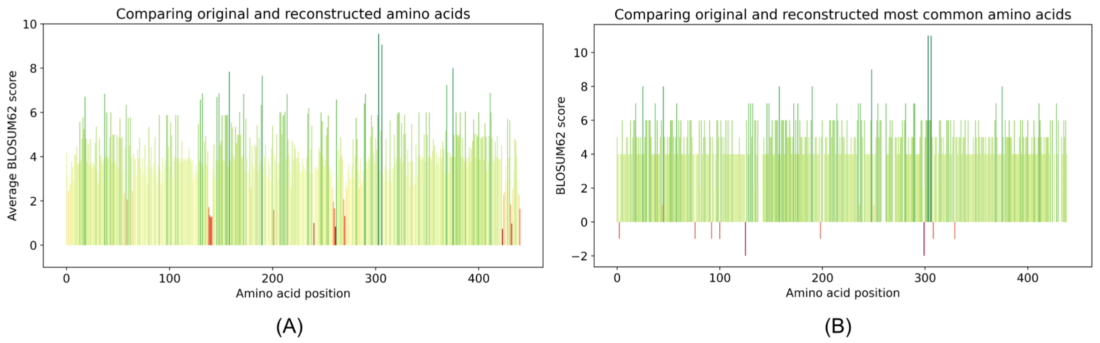

Figure 4: (A) Amino acid reconstruction performance using average BLOSUM62 score for all the sequences of one protein family (Enolase). Each bar shows the average BLOSUM62 score per position in the protein sequence. (B) Comparing the most common amino acid in the original dataset with the most common amino acid of the reconstructed alignment using the BLOSUM62 score for the Enolase protein family. In 95% of the cases, the reconstructed amino acid is equal to the most common one in the alignment position.

In the next step, since the space of all fingerprints is a 30-dimensional vector space, we tested how different the fingerprints of other families are from one another by using k-means clustering with 10-fold cross-validation. We separate the train and test datasets and calculate the accuracy of the clustering model on the test dataset. Our analysis shows that our variational autoencoder creates unique and distinguishable fingerprints (the latent spaces for each family) for each protein family with 100% accuracy. This indicates that our variational autoencoder perfectly extracts the most important and informative information from the data into a compact representation of 30 floats as fingerprints. It also makes it easy to compare families of different sizes and sequences since they are all represented by vectors of the same length.

3.2 Phase 2 – classification

Performance metrics: We use four metrics to evaluate the performance of our model for each protein family – F1 score, Precision, Recall, and Accuracy, defined as follows:

- TP (True Positive): The number of sequences from family x, predicted as x.
- TN (True Negative): The number of sequences from a family other than x, not predicted as x.
- FP (False Positive): The number of sequences from a family other than x, predicted as x.
- FN (False Negative): The number of sequences from family x, not predicted as x.
- Recall (or Sensitivity) for family x, shows the ratio of correctly predicted sequences for all the sequences of family x: $Recall = \frac{TP}{(TP+FN)}$
- Precision for family x, shows the ratio of correctly predicted sequences for all the sequences predicted as x: $Precision = \frac{TP}{(TP+FP)}$
- $F1\_score$ is the harmonic mean of precision and recall for each family. This score considers both of them in a balanced way: $F1\_score = \frac{(2 \times Recall \times Precision)}{(Recall+Precision)}$

Table 1: Performance comparison of protein family classification

Protein family name	**F1** ([27])	**Precision** ([27])	**Recall** ([27])	**Accuracy**
Class-II aminoacyl-tRNA synthetase family	1.0000 (0.995)	1.0000 (1.000)	1.00 (0.991)	1.0000
Class-I aminoacyl-tRNA synthetase	1.0000 (0.985)	1.0000 (0.987)	1.00 (0.982)	1.0000
Methyltransferase superfamily	1.0000 (0.939)	1.0000 (0.938)	1.00 (0.939)	1.0000
Class I-like SAM-binding methyltransferase	0.9901 (0.941)	0.9804 (0.940)	1.00 (0.932)	0.9996
Protein kinase	1.0000 (0.970)	1.0000 (0.973)	1.00 (0.965)	1.0000
TRAFAC class translation factor GTPase	1.0000 (0.993)	1.0000 (0.994)	1.00 (0.992)	1.0000
ABC transporter	1.0000 (0.995)	1.0000 (1.000)	1.00 (0.991)	1.0000
G-protein coupled receptor 1 family	0.9899 (0.971)	1.0000 (0.987)	0.98 (0.955)	0.9996
Radical SAM	1.0000 (0.983)	1.0000 (0.997)	1.00 (0.970)	1.0000
TRAFAC class TrmE-Era-EngA-EngB-Septin-like GTPase	1.0000 (0.980)	1.0000 (0.979)	1.00 (0.977)	1.0000
Cytochrome b family	1.0000 (0.978)	1.0000 (0.982)	1.00 (0.993)	1.0000
MurCDEF family	1.0000 (0.989)	1.0000 (0.984)	1.00 (0.995)	1.0000
Cytochrome P450	1.0000 (0.986)	1.0000 (1.000)	1.00 (0.974)	1.0000
EPSP synthase	1.0000 (0.984)	1.0000 (0.969)	1.00 (1.000)	1.0000
Transferase hexapeptide repeat	1.0000 (0.977)	1.0000 (0.955)	1.00 (1.000)	1.0000
Universal ribosomal protein uS2	1.0000 (0.994)	1.0000 (0.994)	1.00 (0.994)	1.0000
Universal ribosomal protein uL2	0.9899 (0.994)	1.0000 (0.994)	0.98 (0.994)	0.9996
GHMP kinase	1.0000 (0.961)	1.0000 (0.954)	1.00 (0.967)	1.0000
Universal ribosomal protein uS3	1.0000 (0.984)	1.0000 (1.000)	1.00 (0.979)	1.0000
Heat shock protein 70	1.0000 (0.990)	1.0000 (1.000)	1.00 (0.979)	1.0000
Prokaryotic/mitochondrial release factor	0.9901 (1.000)	0.9804 (1.000)	1.00 (1.000)	0.9996
Enolase	1.0000 (0.986)	1.0000 (0.993)	1.00 (0.979)	1.0000
Short-chain dehydrogenases/reductases (SDR)	0.9899 (0.941)	1.0000 (0.948)	0.98 (0.934)	0.9996
Universal ribosomal protein uL1	1.0000 (0.975)	1.0000 (0.846)	1.00 (0.975)	1.0000
MnmG	1.0000 (0.992)	1.0000 (0.992)	1.00 (0.992)	1.0000
Krueppel C2H2-type zinc-finger protein	1.0000 (0.975)	1.0000 (0.958)	1.00 (0.993)	1.0000
Methylthiotransferase	1.0000 (0.996)	1.0000 (1.000)	1.00 (0.991)	1.0000
Universal ribosomal protein uS4	1.0000 (0.979)	1.0000 (1.000)	1.00 (0.958)	1.0000
NAD-dependent DNA ligase	1.0000 (0.996)	1.0000 (0.993)	1.00 (1.000)	1.0000
TRAFAC class OBG-HflX-like GTPase	0.9899 (0.990)	1.0000 (0.987)	0.98 (0.993)	0.9996
Universal ribosomal protein uL3	1.0000 (0.992)	1.0000 (1.000)	1.00 (0.985)	1.0000
SHMT	1.0000 (0.992)	1.0000 (0.984)	1.00 (0.992)	1.0000
Adenylosuccinate synthetase	1.0000 (0.984)	1.0000 (1.000)	1.00 (0.969)	1.0000
SecA	1.0000 (1.000)	1.0000 (1.000)	1.00 (1.000)	1.0000
Class-III pyridoxal-phosphate-dependent	1.0000 (0.982)	1.0000 (0.973)	1.00 (0.991)	1.0000
FKBP-type PPIase	1.0000 (0.939)	1.0000 (0.931)	1.00 (0.947)	1.0000
RNA polymerase alpha chain	1.0000 (0.967)	1.0000 (0.992)	1.00 (0.944)	1.0000
Tetrahydrofolate dehydrogenase/cyclohydrolase	0.9901 (0.986)	0.9804 (0.986)	1.00 (0.986)	0.9996
EF-Ts	1.0000 (0.988)	1.0000 (0.984)	1.00 (0.992)	1.0000
Universal ribosomal protein uL4	0.9901 (0.988)	0.9804 (0.975)	1.00 (1.000)	0.9996
Polyribonucleotide nucleotidyltransferase	1.0000 (0.996)	1.0000 (1.000)	1.00 (0.992)	1.0000
Glycosyltransferase 28	1.0000 (0.969)	1.0000 (0.964)	1.00 (0.973)	1.0000
RuvB	1.0000 (0.987)	1.0000 (0.991)	1.00 (0.982)	1.0000
Fmt family	1.0000 (0.979)	1.0000 (0.991)	1.00 (0.967)	1.0000
RecA family	1.0000 (0.965)	1.0000 (0.986)	1.00 (0.945)	1.0000
IPP transferase family	1.0000 (0.989)	1.0000 (1.000)	1.00 (0.979)	1.0000
Average	**0.9983 (0.977)**	**0.9983 (0.976)**	**0.9983 (0.977)**	**0.9999**

The numbers mentioned inside the parentheses are the performance results from [27]. The authors of [27] did not provide accuracy results.

- Accuracy is the ratio of correct predictions for all the predictions for each family: $Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)}$

Performance results: We calculate the Precision, Recall, Accuracy, and F1_score for each protein family and also the average values. Table 1 shows the classification performance for 12 randomly selected protein families. The table compares the F1 score, precision, and recall of our model and [27] for each protein family. The accuracy values from [27] are not available. The results show our model performs better than the model used in [27] on average. The average F1_score, Precision, and Recall for our model are all 0.9983, and for their model are 0.977, 0.976, and 0.977, respectively. Zhang et al. [27] compared the performance of their model with other state-of-the-art methods mentioned in [39], which include LSTM, biLSTM, GRU, CNN (Three layers), and SVM. They showed that their model outperforms the best methods. Therefore, we can conclude that our model outperforms these methods.

4 Conclusions

We present a variational autoencoder and a convolutional neural network that work together to classify proteins into families using their amino acid sequences. Our pipeline does not require the structure of the proteins or other information, which makes it easier and faster to use. We use the sequence alignment, predicted secondary structure, and predicted solvent accessibility as inputs to our variational autoencoder to generate a small informative fingerprint. The CNN model then predicts protein families using the compressed fingerprint representation. The results show our variational autoencoder can generate completely distinguishable fingerprints for the families. The CNN model can classify proteins with an average accuracy of 99.99%, which offers better performance than existing methods. Our next step is to test our model on several other datasets and also to create neural network models to predict protein contact maps and binding interfaces using the fingerprints generated by our variational autoencoder. We present open-source tools to use our algorithms and hope that this will spur further advances for automatic protein classification from the community. Our next steps involve comparing our model with transformers like ProBERTA [40] and ESM-1b [41], and including metrics, such as saliency maps, t-SNE visualization, and N-gram features of the datasets.

Acknowledgments

The simulations were run in part on the UMass Boston research cluster. Part of the research was funded by a UMass Boston Proposal development grant (NH and DH).

References

[1] D. H. Haft. TIGRFAMs: a protein family resource for the functional identification of proteins. *Nucleic Acids Research*, 29(1):41–43, January 2001.

[2] Cathy H Wu, Hongzhan Huang, Lai-Su L Yeh, and Winona C Barker. Protein family classification and functional annotation. *Computational Biology and Chemistry*, 27(1):37–47, February 2003.

[3] Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors. *Encyclopedia of Bioinformatics and Computational Biology - Volume 1.* Elsevier, 2019.

[4] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, July 2021.

[5] Ewen Callaway. What's next for alphafold and the ai protein-folding revolution, Apr 2022.

[6] Burkhard Rost. Twilight zone of protein sequence alignments. *Protein Engineering, Design and Selection*, 12(2):85–94, February 1999.

[7] L Holm. The FSSP database: fold classification based on structure-structure alignment of proteins. *Nucleic Acids Research*, 24(1):206–209, January 1996.

[8] Bo Liu, Theodore Gibbons, Mohammad Ghodsi, Todd Treangen, and Mihai Pop. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. *BMC Genomics*, 12(S2), July 2011.

[9] Stephen Nayfach, Patrick H. Bradley, Stacia K. Wyman, Timothy J. Laurent, Alex Williams, Jonathan A. Eisen, Katherine S. Pollard, and Thomas J. Sharpton. Automated and accurate estimation of gene family abundance from shotgun metagenomes. *PLOS Computational Biology*, 11(11):e1004573, November 2015.

[10] Ehsaneddin Asgari and Mohammad R.K. Mofrad. Deep genomics and proteomics: Language model-based embedding of biological sequences and their applications in bioinformatics. In *Leveraging Biomedical and Healthcare Data*, pages 167–181. Elsevier, 2019.

[11] Karl Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, November 1901.

[12] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, February 1991.

[13] Gabriela Czibula, Alexandra-Ioana Albu, Maria Iuliana Bocicor, and Camelia Chira. AutoPPI: An ensemble of deep autoencoders for protein–protein interaction prediction. *Entropy*, 23(6):643, May 2021.

[14] Huaming Chen, Jun Shen, Lei Wang, and Jiangning Song. Leveraging stacked denoising autoencoder in prediction of pathogen-host protein-protein interactions. In *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, June 2017.

[15] Yan-Bin Wang, Zhu-Hong You, Xiao Li, Tong-Hai Jiang, Xing Chen, Xi Zhou, and Lei Wang. Predicting protein–protein interactions from protein sequences by a stacked sparse autoencoder deep neural network. *Molecular BioSystems*, 13(7):1336–1344, 2017.

[16] Leilei Wang and Jinyong Cheng. Protein secondary structure prediction using AutoEncoder network and bayes classifier. *IOP Conference Series: Materials Science and Engineering*, 322(6):062008, March 2018.

[17] Richa Dhanuka, Anushree Tripathi, and Jyoti P. Singh. A semi-supervised autoencoder-based approach for protein function prediction. *IEEE Journal of Biomedical and Health Informatics*, 26(10):4957–4965, October 2022.

[18] Mihai Teletin, Gabriela Czibula, Maria-Iuliana Bocicor, Silvana Albert, and Alessandro Pandini. Deep autoencoders for additional insight into protein dynamics. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 79–89. Springer International Publishing, 2018.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv, 2013.

[20] Qiuyu Zhu, Hu Wang, and Ruixin Zhang. Wavelet loss function for auto-encoder. *IEEE Access*, 9:27101–27108, 2021.

[21] Fardina Fathmiul Alam and Amarda Shehu. Variational autoencoders for protein structure prediction. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. ACM, September 2020.

[22] Alex Hawkins-Hooker, Florence Depardieu, Sebastien Baur, Guillaume Couairon, Arthur Chen, and David Bikard. Generating functional protein variants with variational autoencoders. *PLOS Computational Biology*, 17(2):e1008736, February 2021.

[23] Xiaojie Guo, Yuanqi Du, Sivani Tadepalli, Liang Zhao, and Amarda Shehu. Generating tertiary protein structures via interpretable graph variational autoencoders. *Bioinformatics Advances*,

1(1), January 2021.

[24] Hao Tian, Xi Jiang, Francesco Trozzi, Sian Xiao, Eric C. Larson, and Peng Tao. Explore protein conformational space with variational autoencoder. *Frontiers in Molecular Biosciences*, 8, November 2021.

[25] Alexandra-Ioana Albu. Towards learning transferable embeddings for protein conformations using variational autoencoders. *Procedia Computer Science*, 192:10–19, 2021.

[26] Joe G. Greener, Lewis Moffat, and David T Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific Reports*, 8(1), November 2018.

[27] Da Zhang and Mansur R. Kabuka. Protein family classification from scratch: A CNN based deep learning approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(5):1996–2007, September 2021.

[28] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[29] Agnes Toth-Petroczy, Perry Palmedo, John Ingraham, Thomas A. Hopf, Bonnie Berger, Chris Sander, and Debora S. Marks. Structured states of disordered proteins from genomic sequences. *Cell*, 167(1):158–170.e12, September 2016.

[30] Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence- and structure-rich era. *Proceedings of the National Academy of Sciences*, 110(39):15674–15679, September 2013.

[31] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. Bias in machine learning software: why? how? what to do? In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, August 2021.

[32] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, December 2011.

[33] J. Cheng, A. Z. Randall, M. J. Sweredoski, and P. Baldi. SCRATCH: a protein structure and structural feature prediction server. *Nucleic Acids Research*, 33(Web Server):W72–W76, July 2005.

[34] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132:377–384, 2018.

[35] Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. Medical image classification with convolutional neural network. In *2014 13th International Conference on Control Automation Robotics and Vision (ICARCV)*. IEEE, December 2014.

[36] S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks., Nov 1992.

[37] Sean R Eddy. Where did the BLOSUM62 alignment score matrix come from? *Nature Biotechnology*, 22(8):1035–1036, August 2004.

[38] William R. Pearson. Selecting the right similarity-scoring matrix. *Current Protocols in Bioinformatics*, 43(1), October 2013.

[39] Timothy Lee and T. Nguyen. Protein family classification with neural networks. 2016.

[40] Ananthan Nambiar, Maeve Heflin, Simon Liu, Sergei Maslov, Mark Hopkins, and Anna Ritz. Transforming the language of life, Sep 2020.

[41] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences, Apr 2021.