CS612 Proposed Project – Classifying Proteins Using Neural Networks and Compressed Protein Representation

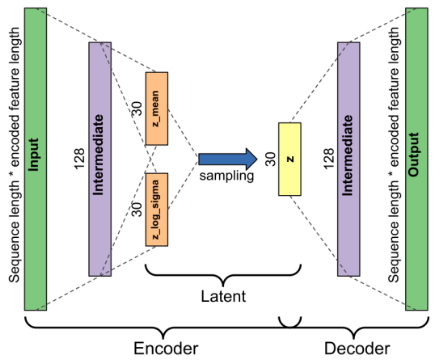
Description

This is a project for two people. The overall goal is to classify proteins into families using autoencoders and neural networks. For example, see here:
`https://mpsych.org/papers/dehghanpoor2023classifying.pdf`
(The full paper may be beyond the scope for a project).

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. 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, protein secondary structure prediction, protein function prediction, protein dynamics studies and many more.

Here is an illustration:



Encoding Protein Sequences

In general, protein sequences are represented by using twenty letters of amino acid alphabet, while such representation cannot be directly processed before it is converted to digital representation. Obtaining the digital representation of amino acid is the first step of machine-learning based protein structure and function prediction methods, and effective digital representation is crucial to the final success of these methods. The amino acid encoding represents each amino acid of a protein sequence by using different n-dimensional vectors, thus its vector space for a protein sequence is $n \times L$ (L is the length of protein sequence). By combining with different machine learning methods, the amino acid encoding can be used in protein properties prediction both at residue-level and sequence-level (i.e., protein fold recognition, secondary structure prediction, etc). The most widely used encodings are the one-hot encoding, the position specific scoring matrix (PSSM) encoding, and some physico-chemical properties encodings. The binary encoding methods use multidimensional binary digits (0 and 1) to represent amino acids in protein sequences. The most commonly used binary encoding is the one-hot encoding. Specifically, the twenty standard amino acids are fixed in a specific order, and then the ith amino acid type is represented by twenty binary bits with the i^{th} bit set to 1 and others to 0. There is only one bit equal to "1" for each vector, hence it is called "one-hot". For example, the twenty standard amino acids are sorted as [A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y], the one-hot code of A is `100000000000000000000`, C is `01000000000000000000`, and so on. Notice that the vectors are of size 21 and not 20, since an extra bit is needed to represent a gap or an unknown amino acid. Because the one-hot encoding is a high-dimensional and sparse vector representation, there is a simplified binary encoding method based on conservative replacements through evolution. Deriving from the point accepted mutation (PAM) matrices [21], the twenty standard amino acids are divided into six groups: [H, R, K], [D, E, N, Q], [C], [S, T, P, A, G], [M, I, L, V], and [F, Y, W], and six dimensional binary vectors are used to represent amino acids based on their groups. Another low-dimensional binary encoding scheme is the binary 5-bit encoding introduced by White and Seffens [22]. Theoretically, the binary 5-bit code could represent 32 ($2^5 = 32$) possible amino acid types. In order to represent the twenty standard amino acids, the all zeros encoding, the all ones encoding and those encodings with 1 or 4 ones (5+5=10) are removed, finally 20 encodings (32-1-1-10=20) are left in total.

See here for a comprehensive review: `https://ieeexplore.ieee.org/document/8692651`

Neural Networks

The AutoEncoder in itself produces a reduced representation of protein sequences. This representation can be used for feature construction and selection for learning or classification tasks. We mentioned neural networks in class, albeit briefly. Nowadays there are many types of neural networks. Simple(ish) examples include for example – a 1D or 2D CNN (Convolutional neural network). A CNN has one or more convolutional layers, optionally pooling layers to reduce the dimension, fully connected layers and backpropagation for learning and training. Many implementations of CNN exist using keras, tensorflow, pytorch etc. see here:

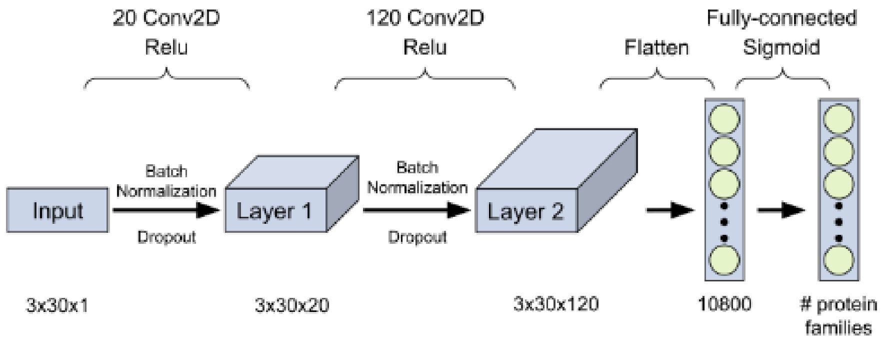
`https://www.tensorflow.org/tutorials/images/cnn`

`https://www.kaggle.com/code/kanncaa1/convolutional-neural-network-cnn-tutorial`

`https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-`

`https://pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network`

Please use an existing code as basis and don't re-invent the wheel.



The Assignment

Construct an auto-encoder for protein multiple sequence alignments. In particular, your autoencoder should do the following:

- Input for first step: A multiple sequence alignment (examples will be provided).
- The AE should be able to encode a latent space (of a small size, anything between 5 and 30, up to you), and decode the sequences back.
- You should measure the reconstruction error by the following parameters:
  1. The learning error.
  2. The ability to reconstruct the same amino acid in each position
  3. The ability to reconstruct the most common amino acid in each position in the alignment.
- Input for second step: The latent spaces from the previous stage.
- The network should accept your input and predict a protein family.
- The nature and architecture of the network is up to you. You may try several combinations of architectures and parameters.

You can choose whatever encoding you want and whatever parameters you want for the autoencoder. To get you started, here are some examples using python. These examples take the MNIST database (handwritten digits). You should be able to change it to protein sequences.

Then, feed the resulting features to a neural network and test the classification ability.

See here for a (not so successful) example, but quite detailed:

`https://medium.com/@rwalroth89/neural-nets-for-protein-classification-1b122cde5b6f`

This example doesn't use autoencoders but just representations of sequences as input features. An interesting test would be to test the features as given in the link above vs. autoencoders and test the difference in performance, and/or use different architectures.

Deliverables

- The code.
- A 3-4 page document outlining the following:
  - A brief introduction to the problem (a paragraph or so).
  - Implementation details: Programming language, encoding method, choice of encoder(s), parameters of the autoencoder(s) - layers, activation methods, latent space dimensions.
  - Results: Reconstruction error, example of at least one MSA. You can use the plotting functions presented in the links above.
  - The Neural network parameters, choice of architecture, classification ability.