# CS612 Term Project - Autoencoders for Protein Sequences

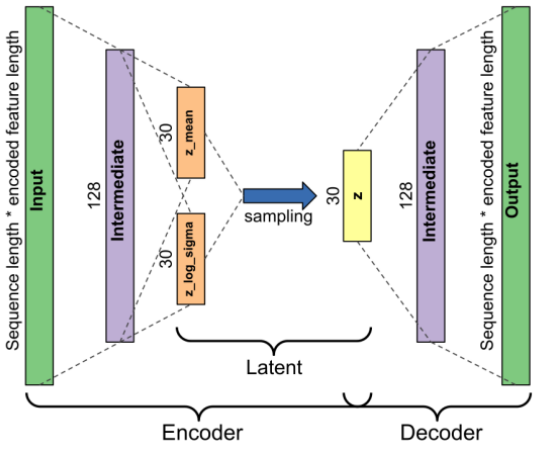
## Auto Encoders (AE)

Autoencoders are unsupervised feed-forward artificial neural net- works 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 dimen- sion 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 net- work 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.

## 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), 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 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 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. Some applications of VAE in protein studies include protein structure prediction, generating protein variants and tertiary structures, protein conformational space exploration, protein fold design, and many others.

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 `10000000000000000000`, 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`

# The Assignment

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

- Input: 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 32, up to you), and decode the sequences back.
- You should measure the reconstruction error by the following parameters:
  1. The learning error (cross-entropy, kl divergence, etc.).
  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.
- 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 can just copy over the code to get you started. You should then be able to adapt it to protein sequences.

## Suggested Steps

I suggest you break down the assignment into several smaller steps:

- Run the given examples below, make sure you understand them.
- Implement utilities that will help you with proteins:
  - Read FASTA format files
  - Implement encoding (one hot or others)
  - Understand the autoencoder parameters, at least in general (read the Keras documentation, or whatever engine you use).
  - Figure out how to decode and retrieve a reconstructed sequence alignment.
  - Compare original and reconstructed sequences.

### A little extra credit

Build a variational autoencoder and compare your results with a regular autoencoder. The comparison should include the learning rate, the reconstruction error and the result classification. You don't necessarily have to use Keras (or python at all), but I found it easy to use

Autoencoders vs. Variational autoencoders:
`https://blog.keras.io/building-autoencoders-in-keras.html`
`https://blog.paperspace.com/how-to-build-variational-autoencoder-keras/`
Here are some more resources:
`https://towardsdatascience.com/how-to-make-an-autoencoder-2f2d99cd5103`
`https://www.kaggle.com/code/samlinz/simple-mnist-autoencoder`

## Deliverables

- The code.
- A 1-2 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.