Exercise 8
Neural Networks
Machine Learning Workflow

1 Pre-Processing
   ▶ Parse raw data
   ▶ One-Hot-Encoding of sequences
   ▶ Create sliding window view for each sample
   ▶ Encode class labels as integers
   ▶ Clean up data, e.g. remove unresolved residues
   ▶ Split data set into a training and a test set

2 Data Analysis
   ▶ Number of samples in the final data set
   ▶ Number of unresolved residues

3 Training the Network
   ▶ Implement forward and backward pass
   ▶ (Optimize meta-parameters, s.a. number of neurons, epochs..)

4 Performance Evaluation
Task: Classify residues in a transmembrane-helix from all other residues. Using one-hot-encoding of amino acid letters and a sliding window approach to take neighbouring residues into account.

Data:

Possible labels:
- 0: non-membrane (inside)
- 1: non-membrane (outside)
- 2: non-membrane (unknown topology)
- L: membrane re-entrant loop
- U: unknown/unresolved
- H: transmembrane helix
- h: transmembrane helix
Encode all residues in a protein via one-hot-encoding and create a sliding window view for each residue. Here, we use zero-padding to add elements to the start and end of a protein in order to not 'loose' these residues.
Network Architecture

Rectified Linear Unit (ReLU): \( y(x) = \max(0, x) \)

Softmax: \( \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \) for \( j = 1, \ldots, K \)

Numerically stable softmax: \( \sigma(z)_j = \frac{e^{z_j - \max(z)}}{\sum_{k=1}^{K} e^{z_k - \max(z)}} \)

Cross-Entropy loss: \( H(y, y_{\text{pred}}) = - \sum_i y_i \log(p_i) \)

Delta Cross-Entropy with softmax output: \( \frac{\partial L}{\partial o_i} = y_{\text{pred}} - y \)
Online vs. Mini-Batch vs. Batch Learning

- **Online Learning** (weights are updated per sample)
  Update rule: \( W_{ij} = W_{ij} - \mu \cdot \delta_j \cdot X_i \)
  for all epochs
  for all samples
  forward pass
  update weights

- **Batch Learning**: weights are updated after all training samples were processed.
  Update rule: \( W_{ij} = W_{ij} - \mu \cdot \sum_k \delta_j \cdot X_i \) with \( k \) being the number of samples in the batch
  for all epochs
  for all samples
  forward pass
  accumulate gradients
  update weights

- **Mini-Batch learning**: weights are updated after a subset of all training samples was processed. Sum over deltas are usually normalized by number of samples in the mini-batch.
Hints

▶ We apply mini-batch learning with one mini-batch consisting of one protein. The gradients are normalized by dividing them by the number of samples in the minibatch. The normalization is applied at the end of the _delta_cross_entropy function. However, the summation can be done during the backpropagation via dot products of the gradients and weights.

▶ Add a bias term to the input layer as well as to the hidden layer.

▶ Think of the target labels [0,1] as an array of indices. As you have two output units, describing a probability for each class (after softmax normalization), each label [0,1] can be treated as an index for the true class label of a sample. This allows to simplify the summation of the cross entropy loss as one term cancels out (multiplication with 0). The same holds true for the delta of the cross entropy.