Secondary Structure Relative Solvent Accessibility Prediction from raw MSA Data

- **Who**: Names and logins of all your group members.

Sophia Li: sli347

Ethan Ye: eye5

Kelley Tu: kjtu

Russell Chiu: rhchiu

- **Introduction**: What problem are you trying to solve and why?
    - If you are implementing an existing paper, describe the paper's objectives and why you chose this paper.
        - The objective of this paper is to use deep networks to refine multiple sequence alignments and extract more information from them, specifically protein structure features. Specifically, we were interested in NLP techniques used in computational biology. The article implements two different architectures: one for SS-RSA (secondary structure and relative solvent accessibility predictions) and one for CMAP (contact map prediction).
            - SS-RSA is the prediction of the secondary structure and relative solvent accessibility. The SS-RSA model uses Long Short-Term Memory (LSTM), which is a specialized type of recurrent neural network (RNN) particularly well-suited for processing sequential data in Natural Language Processing (NLP). The master protein sequence is treated like a sentence. Each residue's environment (in an MSA window) is like a word with context. The LSTM layer learns the contextual relationships between residues (e.g., how one residue's environment influences the structure of another).
            - Protein contact maps represent the distance between amino acid residue pairs of a 3d structure using a 2d matrix/image. Just as sentences are sequences of words, multiple sequence alignments (MSAs) are treated as sequences of amino acid "tokens." Each amino acid (including gaps and rare codes like X, B, Z) is embedded to a learned vector similar to how words are in the embedding space.
    - What kind of problem is this? Classification? Regression? Structured prediction? Reinforcement Learning? Unsupervised Learning? etc.
        - The SS-RSA model is addressing a supervised classification problem. Each residue in the protein sequence is assigned a label (ex: Helix, Strand, or Coil), and the model performs sequence labeling—a type of structured prediction—where the outputs are discrete classes learned via training with

methods such as LSTM and softmax layers using a categorical cross-entropy loss function. This formulation is common in natural language processing tasks and has been adapted here for predicting structural features of proteins.
        - The CMAP model is a supervised structured prediction system that outputs a full protein contact map—a two-dimensional matrix (or image) representing the pairwise distances between amino acid residues in a 3D structure. The network applies several convolutional layers to extract features from these embedded sequences. A custom layer performs an outer product operation on the convolutional features to model interactions between all residue pairs, effectively generating a four-dimensional tensor that is reshaped into an L × L matrix (with L being the length of the protein). Subsequent convolutional layers and a final softmax layer are used to classify each residue pair—producing a probability that indicates whether a contact exists between them.

- **Related Work**: Are you aware of any, or is there any prior work that you drew on to do your project?
    - Please read and briefly summarize (no more than one paragraph) at least one paper/article/blog relevant to your topic beyond the paper you are re-implementing/novel idea you are researching.
        - Protein secondary structure describes the local conformation of a protein's polypeptide backbone, determined by characteristic hydrogen-bonding patterns between portions of the polypeptide sequence. The two most common regular motifs are α-helices, in which local segments of the polypeptide form coiled structures, and β-sheets, in which the polypeptide folds into a pleated pattern. Irregular regions with no stable hydrogen-bonding pattern are grouped as coil or loop segments. DSSP assigns eight precise structural states (H, G, I for helices; E, B for sheets; and T, S, L for turns and bends), which are conventionally reduced to three classes: helix, sheet, and coil. These secondary elements serve as fundamental building blocks for the overall three-dimensional fold of the protein.
    - In this section, also include URLs to any public implementations you find of the paper you're trying to implement. Please keep this as a "living list"–if you stumble across a new implementation later down the line, add it to this list.
        - Paper: https://www.biorxiv.org/content/biorxiv/early/2018/10/22/394437.full.pdf

- **Data**: What data are you using (if any)?
    - If you're using a standard dataset (e.g. MNIST), you can just mention that briefly. Otherwise, say something more about where your data come from (especially if there's anything interesting about how you will gather it).
        - The dataset used in the paper consists of multiple sequence alignments (MSAs) representing protein sequences, which were obtained with HHblits and

JackHMMER. We plan to obtain additional testing data from here by gathering MSAs from randomly selected protein domains.

- How big is it? Will you need to do significant preprocessing?
  - In the original paper, the dataset included 29,653 protein chains; however, due to limited computational resources, we plan to scale down the dataset during our own training.
  - For our additional testing data, we need to convert the format of the protein sequences from Stockholm to Fasta. Otherwise, no other preprocessing should be necessary as the model will take in rawMSA data.

- **Methodology**: What is the architecture of your model?
  - How are you training the model?
    - In the paper, they rigorously trained the model on a large set of proteins (for the SS-RSA model, they trained 100-1000 sequences for 5 epochs), but because of time constraints and less compute power, we are going to train on a subset of all the training data for 5 epochs.
    - We are going to use Oscar, Kaggle Notebooks, or Hydra because we will need GPUs to train the model, as the MSA datasets are relatively large. For reference, the researchers of the paper we are implementing used computers equipped with NVIDIA GeForce 1080Ti, Tesla K80, and Quadro P6000 GPUs.
  - If you are implementing an existing paper, detail what you think will be the hardest part about implementing the model here.
    - The MSA data they trained the model on is inaccessible and extremely large (150 GB), so we need to find a way to extract this data.
    - In the repo, they have all the TensorFlow/Keras .h5 models, but because .h5 files are stored in binary, they don't contain readable source code, so we will have to reverse engineer to find the layer-by-layer breakdown/implementation.
    - Due to the sheer volume of data and our limited access to GPUs, we will need to keep the complexity of the model down so that we can train in a reasonable amount of time.

- **Metrics**: What constitutes "success?"
  - What experiments do you plan to run?
    - We plan to experiment with the number of layers as paper implements a range of 2d convolution/pooling layers.
  - For most of our assignments, we have looked at the accuracy of the model. Does the notion of "accuracy" apply for your project, or is some other metric more appropriate?
    - High degrees of accuracy will be hard to achieve, but it forms the best metric for our model's success by comparing it to other relevant sequencing prediction models.

    - If you are implementing an existing project, detail what the authors of that paper were hoping to find and how they quantified the results of their model.
        - The authors of the rawMSA paper set out to show that supplying a raw multiple sequence alignment directly to a deep network could surpass or match classical methods.
    - What are your base, target, and stretch goals?
        - Base goal: To develop a model that can make some prediction about the secondary structure relative solvent accessibility (SS-RSA) of a protein, in which accuracy is NOT a focus
        - Target Goal: To develop a model that can make decently accurate predictions (greater than 60%) about the SS-RSA of test proteins
        - Stretch Goal: To apply the model in a biological context by enabling it to draw conclusions on the relationship between certain organisms/a family of proteins

- **Ethics**: Choose 2 of the following bullet points to discuss; not all questions will be relevant to all projects so try to pick questions where there's interesting engagement with your project. (Remember that there's not necessarily an ethical/unethical binary; rather, we want to encourage you to think critically about your problem setup.)
    - Why is Deep Learning a good approach to this problem?
        - Deep learning is a good approach to this problem because protein data is large, and classic methods for protein structure prediction rely on manually engineered features. With deep learning, it eliminates the manual step and allows for learning directly from raw MSAs. Embedding layers allow the network to learn context-aware representations of amino acids. LSTM helps with learning long range residue dependencies to help predict structure.
    - What broader societal issues are relevant to your chosen problem space?
        - Protein structure predictions are key to understanding the underlying mechanisms behind diseases like Alzheimer's and Parkinson's, which are caused by protein misfolding. Accurate structural models may be able to help identify drug targets and guide drug design for such diseases. By creating a model that uses raw MSA data (without the need for significant manual preprocessing), this implementation enables more efficient and scalable predictions that may facilitate advancements in therapeutics.

- **Division of labor**: Briefly outline who will be responsible for which part(s) of the project.
    - Kelley: data collection
    - Sophia: architecture evaluation
    - Russell: architecture evaluation
    - Ethan: data collection
    - Everyone: poster, implementation

```
2.15.0
['models_ss/epoch_2_window_31_depth_1000_stage1_1_convol_10_pool_20_stage2_2_bidir_350_drop0.4_fold_3_val_0.8283165404861786_q3_0.8186187038689269.h5', 'models_ss/epoc
h_2_window_31_depth_200_stage1_1_convol_10_pool_20_stage2_2_bidir_350_drop0.5_fold_0_val_0.8274012059086158_q3_0.8096935512140271.h5']
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 31000)]           0

 embedding_1 (Embedding)     (None, 31000, 14)         364

 reshape_1 (Reshape)         (None, 31, 1000, 14)      0

 conv2d_1 (Conv2D)           (None, 31, 1000, 14)      1974

 activation_1 (Activation)   (None, 31, 1000, 14)      0

 max_pooling2d_1 (MaxPoolin  (None, 31, 50, 14)        0
 g2D)

 reshape_2 (Reshape)         (None, 31, 700)           0

 bidirectional_1 (Bidirecti  (None, 31, 350)           2942800
 onal)

 dropout_1 (Dropout)         (None, 31, 350)           0

 bidirectional_2 (Bidirecti  (None, 31, 350)           1962800
 onal)

 dropout_2 (Dropout)         (None, 31, 350)           0

 flatten_1 (Flatten)         (None, 10850)             0

 dense_1 (Dense)             (None, 50)                542550

 dropout_3 (Dropout)         (None, 50)                0

 dense_2 (Dense)             (None, 20)                1020

 dropout_4 (Dropout)         (None, 20)                0

 dense_3 (Dense)             (None, 3)                 63

=================================================================
Total params: 5451571 (20.80 MB)
Trainable params: 5451571 (20.80 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
input_1 InputLayer {'batch_input_shape': (None, 31000), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_1'}
embedding_1 Embedding {'name': 'embedding_1', 'trainable': True, 'dtype': 'float32', 'batch_input_shape': (None, None), 'input_dim': 26, 'output_dim': 14, 'embeddings_
initializer': {'module': 'keras.initializers', 'class_name': 'RandomUniform', 'config': {'minval': -0.05, 'maxval': 0.05, 'seed': None}, 'registered_name': None}, 'emb
eddings_regularizer': None, 'activity_regularizer': None, 'embeddings_constraint': None, 'mask_zero': False, 'input_length': None}
```