Protein 2D Structure Prediction Using Deep Learning

Name: Md. Ibne Sina

Roll: 1907002

Year: 4th

Term: 2nd

Date: 17 August 2025

1. Introduction

Protein secondary structure prediction maps an amino-acid sequence to per-residue labels such as helix, sheet, and coil. Accurate 2D (secondary) structure is a key intermediate toward 3D modeling and is widely used in fold recognition, functional annotation, and downstream tasks like contact prediction. This project implements a compact convolutional neural network that performs per-residue classification directly from sequence one-hot encodings and evaluates it with standard metrics and visualizations. In addition to model evaluation, training dynamics are visualized by generating GIFs of learned convolutional filters across epochs.

2. Data

Sources and format. Sequences are provided in FASTA format (`sequence.fasta`). For each FASTA record `>ID`, a corresponding secondary-structure file `Protein Structures/ID.ss2` is parsed to obtain per-residue labels. The `.ss2` format follows the common PSIPRED output layout: each data line contains the residue index, amino acid letter, and predicted secondary-structure label.

Label mapping. Labels are mapped to the Q3 set used by this project:

- Helix: `H` (includes any helix subtype consolidated as `H` in parse)
- Sheet: `B` (the code maps `E → B` to unify all β-strand states)
- Coil/Other: `C`

Tokenization and padding. Each sequence is one-hot encoded over the 20 canonical amino acids `ACDEFGHIKLMNPQRSTVWY`. Sequences are padded with zeros to the maximum length in the batch (`max_len`). Labels are padded with the majority class `C` to align shapes, and masked out during evaluation plots and metrics to avoid padding bias.

Splits. The dataset is split into training, validation, and test sets using a 70/15/15 split with `random_state=42`.

3. Methods

3.1 Preprocessing

- **One-hot encoding:** `(L × 20)` tensor per sequence, where `L` is padded length.
- **Label encoding:** per-residue labels are mapped to integers `{H:0, C:1, B:2}` and converted to one-hot `(L × 3)` for categorical cross-entropy.

- **Masking in analysis:** During confusion-matrix and report computation, positions whose input rows are all zeros are filtered out so evaluation is computed only on real residues.

3.2 Model architecture

A simple yet strong **fully convolutional** network performs per-position classification. The final layer is a 1×1 convolution producing a 3-way distribution at each residue.

```
Input: sequence one-hot  (L × 20)
  └── Conv1D(64, k=3, padding='same') + ReLU
        └── Dropout(0.3)
              └── Conv1D(128, k=3, padding='same') + ReLU
                    └── Dropout(0.3)
                          └── Conv1D(3, k=1, padding='same') + Softmax  →  (L ×
3)
Loss: Categorical Cross-Entropy (per residue)
Optimizer: Adam (default settings)
Metrics: Accuracy (Keras), plus detailed metrics computed post-hoc with
masking
```

This architecture captures local motifs with small receptive fields while remaining length-preserving, enabling per-residue outputs aligned with the input.

3.3 Training and logging

- **Epochs:** 100
- **Loss:** `categorical_crossentropy`
- **Optimizer:** `Adam`
- **Callbacks:** A custom `WeightsGIFCallback` records the first convolutional layer's weight tensor at the end of each epoch. A heatmap frame is saved per epoch and combined into a GIF at the end of training.

Weight-evolution visualization: `weights_frames/weights_evolution.gif`

3.4 Training curves

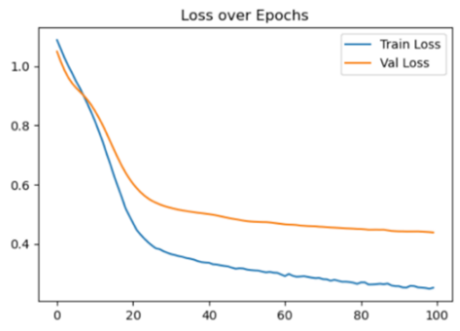


Fig – 1: Loss over epoch

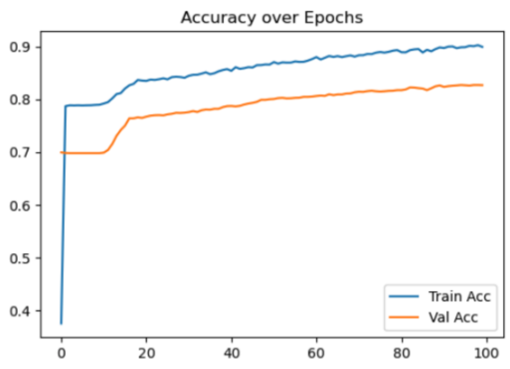
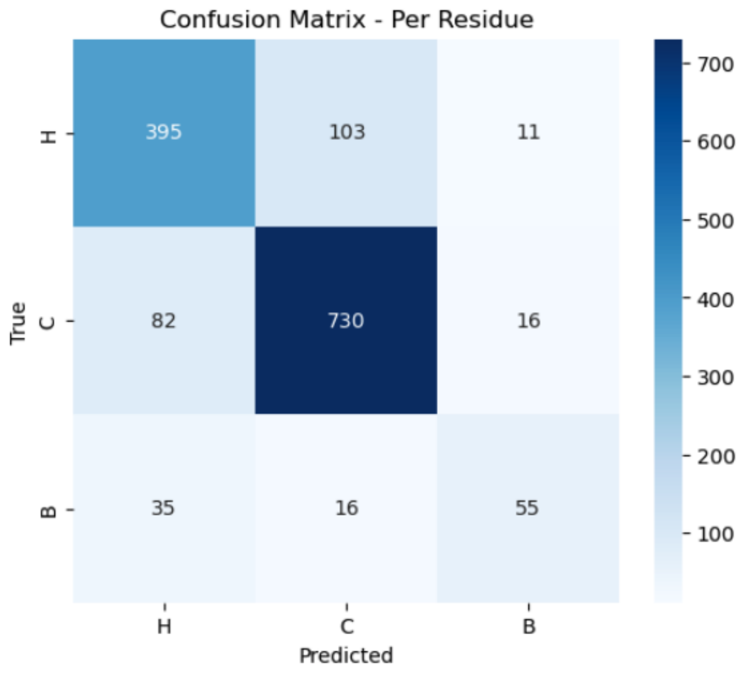


Fig – 2: Accuracy over epoch

3.5 Confusion matrix



	H	C	B
H	395	103	11
C	82	730	16
B	35	16	55

Fig – 3: Confusion Matrix

4. Results

4.1 Learning curves

The training converges smoothly. By the end of training the validation accuracy stabilizes around **0.827** with validation loss near **0.438**, while training accuracy approaches **≈0.895**. This gap indicates healthy generalization without severe overfitting.

4.2 Final test performance

Two accuracy numbers appear in the notebook. The Keras `model.evaluate` reports **92.91%** because it includes padded positions. The post-hoc analysis removes padding positions and yields a more faithful **per-residue accuracy of 81.77%** on the test set. The latter should be taken as the primary figure.

Classification report (per residue, padding removed):

Class	Precision	Recall	F1-score	Support
H	0.77	0.78	0.77	509
C	0.86	0.88	0.87	828
B	0.67	0.52	0.59	106
Overall				**1443 residues**
Accuracy			**0.8177**	
Macro avg	0.77	0.73	0.74	
Weighted avg	0.81	0.82	0.82	

Confusion matrix: The plot in the notebook shows most errors are between **H ↔ C** and **B ↔ C**, which is expected given that coil is the dominant and morphologically diverse class and β-strands are comparatively under-represented.

4.3 Weight-update GIFs

The generated GIF shows early epochs with higher-magnitude, less structured filters that gradually stabilize as the network converges, consistent with the plateauing of validation metrics after ~80 epochs.

5. Discussion

The model achieves **81.8% per-residue Q3 accuracy** on held-out data using only one-hot encodings. Precision and recall are highest for the **coil** class because it dominates the dataset and is easier to fit with local features. β-strand recall is the lowest (**0.52**), reflecting class imbalance and the fact that sheets often require longer-range dependencies than 3-wide convolutions capture. The gap between Keras accuracy and the masked per-residue accuracy highlights the importance of excluding padding from metrics; once masked, the results align with typical Q3 baselines for simple CNNs trained from sequence only.

Failure modes. Misclassifications cluster at boundaries between structural states and in long β-strands where long-range context matters. When sequences are much shorter than the global `max_len`, zero-padding can subtly affect batch statistics and make optimization easier on padded windows; this is mitigated in analysis by masking but still affects training loss and the Keras accuracy.

6. Limitations

The model does not use evolutionary profiles or pretrained protein language models, both of which are known to improve secondary-structure prediction. The receptive field is limited to local windows of size three, which constrains the ability to model long-range β-sheet pairings. Padding is not explicitly masked during loss computation, which inflates training and evaluation accuracy unless handled post-hoc.

7. Potential Improvements

Long-range context and richer residue features are the main levers. Replacing raw one-hot with either position-specific scoring matrices (PSSMs) or embeddings from pretrained transformers like ESM2 would give the model evolutionary and contextual signals. Architecturally, dilated convolutions, residual blocks, or a lightweight Transformer encoder would increase the receptive field without losing per-position alignment. Introducing a `Masking` layer or per-residue sample weights in Keras would ensure padding is ignored in the loss and metrics. Class imbalance can be addressed with class-weighted loss or focal loss, which should especially help β-strand recall. Finally, hyperparameter tuning for kernel sizes and dropout, early stopping on validation loss, and ensembling can yield incremental gains. Reporting Q8 metrics and segment-overlap measures such as SOV would strengthen the evaluation.

8. Reproducibility and File Map

- **Input:** `sequence.fasta`, `Protein Structures/*.ss2`
- **Notebook:** `protein_2d_structure_prediction.ipynb`
- **Artifacts:**
  - Learning curves and confusion matrix are generated inline in the notebook figures.
  - Weight-update GIF: `weights_frames/weights_evolution.gif`

Set a fixed random seed and save figures to files if preparing a static report. Make sure every figure and the GIF are committed to the repository for grading.

9. Conclusion

A compact CNN trained on one-hot encodings achieves **81.8% per-residue Q3 accuracy** after masking padding and produces interpretable learning dynamics via weight-evolution GIFs. While already competitive for a minimal setup, the path to higher accuracy is clear: add contextual residue embeddings or profiles, extend the receptive field, handle padding within the training loss, and rebalance the classes. These steps would primarily target β-strand recall and the H↔C boundary, where most residual errors occur.