**NANYANG TECHNOLOGICAL UNIVERSITY**

# SC4001 Group Project:

## Protein Secondary Structure Prediction (Task B)

| Name | Matric Number |
|---|---|
| Ananthula Harineesh Reddy | U2323714G |
| Natraj Kalingarayar Amogh Sriman Kalinganayar | U2323933F |
| Sunilkumar Hrishikesh | U2323903A |

## Table of Contents

## 1. Introduction

Proteins are essential biological molecules whose functions depend heavily on their three-dimensional shapes. These shapes arise from the way a chain of amino acids folds, beginning with the formation of common local patterns known as secondary structures—α-helices (H), β-sheets (E), and coils (C). Accurately predicting these structures from sequence alone is a key problem in computational biology.

Experimental techniques such as X-ray crystallography and NMR provide detailed structural information but are slow, expensive, and not always practical. This motivates computational approaches that learn the relationship between amino-acid sequences and their secondary structure labels.

This project focuses on building and evaluating deep-learning models that predict the secondary structure of each residue in a protein. We consider two tasks:

1. **Q3 prediction**, which classifies residues into helix (H), sheet (E), or coil (C), and
2. **Q8 prediction**, which uses a more detailed set of eight structural classes.

To achieve this, different model architectures were implemented and compared:

## 2. Related Work

Protein secondary structure prediction has been studied for many years. Early machine-learning methods such as HMMs and SVMs could capture basic patterns but struggled with long-range relationships in protein sequences.

Deep learning improved this. Models like SPIDER3, which use bidirectional LSTMs, showed that looking at the sequence in both directions helps the model understand how nearby residues influence structure. CNN-based models then became common for learning short local motifs, although they still have difficulty with long-range interactions.

More recently, Transformers and pretrained models like ESM have become the most effective approaches. Their self-attention mechanism allows them to compare all positions in the sequence at once, making them better at modelling complex dependencies. Overall, prior work suggests that CNNs learn local patterns, BiLSTMs capture sequence context, and Transformer-based models (including ESM) handle long-range dependencies. This motivates evaluating all four in our project.

## 3. Dataset

### 3.1 Source of data

The dataset used in this project is the 2018-06-06 PDB Intersect Pisces dataset, derived from the Protein Data Bank (PDB). Each entry represents one protein chain and includes:

- seq — amino acid sequence (A–Y)
- sst3 — Q3 secondary-structure labels
- sst8 — Q8 secondary-structure labels
- len — sequence length.

### 3.2 Data Cleaning

Basic filtering was applied to remove sequences containing invalid characters (such as “*”) and any entries with missing or empty labels. Extremely short or duplicate sequences were also removed to maintain dataset quality.

### 3.3 Encoding and Padding

Each amino acid and secondary-structure symbol was mapped to an integer. Since protein chains vary in length, sequences were padded with a special zero value so that models could process them in batches. Padding tokens were later ignored during loss computation.

### 3.4 Splitting

The cleaned dataset was divided into:

- 70% training
- 15% validation
- 15% testing

This ensures that model performance is evaluated on proteins not seen during training.

## 4. Data Preparation

This section explains how the dataset was processed before training the models. The main steps were cleaning the raw sequences, converting them into numerical form, and preparing them for both the standard deep-learning models (CNN, BiLSTM, Transformer) and the ESM-based model.

### 4.1 Dataset Loading and Filtering

The dataset was loaded from a CSV file containing each protein’s amino-acid sequence along with its Q3 and Q8 labels.
Any sequences containing invalid characters (e.g., “*”) or incomplete labels were removed.
The cleaned dataset retained only the required fields: $\mathrm{seq}$, $\mathrm{sst3}$, $\mathrm{sst8}$, and $\mathrm{len}$.

### 4.2 Tokenization and Label Encoding

Since neural networks require numerical input, all sequences and labels were converted to integer format:

- Amino acids were mapped to integers (padding = 0).
- Q3 and Q8 labels were also assigned integer codes, with their own padding values.

Each protein therefore produced three integer sequences: amino acids, Q3 labels, and Q8 labels. These served as inputs and targets for the models.

### 4.3 Train/Validation/Test Split

The cleaned data was split at the protein level into training, validation, and test sets.
Each split included both the raw sequences and their integer-encoded forms.
Keeping proteins intact across splits prevented data leakage and allowed all models, including the ESM pipeline, to use the same partitions.

### 4.4 Padding and Length Handling

Protein sequences vary in length, so they were padded using a special index to create uniform tensors. After padding, the true length of each sequence was recalculated by counting the non-padding positions. This was necessary for models (such as BiLSTMs) that must ignore padded residues during processing. Extremely long sequences were filtered out to control memory usage.

#### 4.5 DataLoader Construction

The padded arrays and lengths were converted into PyTorch tensors and grouped into TensorDatasets for Q3 and Q8.
A simple collate function returned batches in the format: `(inputs, labels, lengths)`
DataLoaders were created for training, validation, and testing, handling batching, shuffling, and efficient loading.

#### 4.6 Mask Coverage

A binary mask was generated for every sequence, marking real residues and hiding padded ones. This mask was used throughout the pipeline: LSTMs relied on true lengths to skip padded tokens, and Transformers used the mask to prevent attention on padding. The loss function also ignored padded labels.
Overall, mask coverage was kept high, ensuring that nearly all training positions came from real amino acids rather than padding. This helped the models focus on meaningful residues and improved training stability.

#### 4.7 ESM-Specific Preparation

For the ESM model, the project also kept a second version of the dataset using raw amino-acid strings, since ESM cannot use integer-encoded sequences.
These sequences were filtered with the same length rule and stored as lists for the train, validation, and test splits.

During ESM processing, each sequence is passed into the pretrained model to generate high-dimensional residue embeddings. These embeddings are then used as inputs for the classifier built on top of ESM.

This setup allowed both the standard models and the ESM-based approach to use consistent, well-prepared data.

## 5. Models used

### 5.1 CNN-Based Model

The CNN model was designed to learn short and medium-range patterns in protein sequences, since many secondary structure features arise from local residue interactions. The model works directly on the integer-encoded sequences created during preprocessing.

#### 5.1.1 Input Representation

Each residue index is first converted into a learnable embedding vector.
A small linear layer then adjusts these embeddings to the number of channels used by the CNN. This prepares the sequence for the convolutional layers that follow.

#### 5.1.2 Residual Convolutional Blocks

The main part of the model is a stack of residual 1D convolution blocks.
Each block contains:

- A 1D convolution
- Batch normalisation
- A ReLU activation
- Dropout
- A residual shortcut connection

These layers allow the network to recognise common local motifs, while the residual connection helps with stability during training.

To expand the model's visibility along the sequence, the convolutions use **dilations.**
This allows the network to capture patterns that span more residues without using far deeper layers.

#### 5.1.3 Output Layer

After the convolution stack, the model applies layer normalisation and dropout.
A final linear layer converts each position's features into class scores, producing one prediction per residue for either Q3 or Q8 classification.

#### 5.1.4 Training Procedure

The CNN uses the padded sequence tensors and corresponding labels.
Key training components include:

- Cross-entropy loss with padding ignored
- Adam-based optimisation
- Learning-rate scheduling based on validation performance
- Early stopping to prevent overfitting
- Mixed-precision training for efficiency

The same architecture is used for both Q3 and Q8 prediction, with only the output layer size adjusted.

#### 5.1.5 Fine Tuning

The CNN was trained from scratch, but several tuning steps were applied to stabilise learning. A small learning rate (1.6e-04) was chosen, and a ReduceLROnPlateau scheduler adjusted the learning rate when validation loss stopped improving. Dropout and weight decay were used to limit overfitting, while early stopping prevented over-training once validation accuracy plateaued. Gradient clipping and mixed-precision training kept updates stable.

Although Optuna was mainly used for exploring hyperparameters in the hybrid model, insights from those trials (such as suitable dropout values and effective learning-rate ranges) also informed the fixed configuration used for the CNN.

## 5.2 BiLSTM-Based Model

The BiLSTM model was used to capture the sequential nature of protein chains. Unlike the CNN, which focuses mainly on local patterns, the BiLSTM is designed to look at the sequence in order and use information from both past and future residues when predicting the structure at each position.

### 5.2.1 Input Representation

As with the CNN, the model starts from the integer-encoded amino acid sequences.
Each residue index is mapped to a learnable embedding vector. This lets the model work in a continuous feature space instead of directly on integer IDs.

The embedded sequence is then passed, along with the true sequence lengths, into the recurrent layer.

### 5.2.2 Bidirectional LSTM Layer

The core of the model is a bidirectional LSTM (BiLSTM). It processes the sequence in two directions:

- One LSTM reads from the N-terminus to the C-terminus (forward).
- Another reads from the C-terminus to the N-terminus (backward).

At each residue, the hidden states from both directions are concatenated.
This means the representation at each position takes into account both preceding and following residues, which is important because secondary structure is often influenced by context on both sides.

To handle padding correctly, the model uses the true sequence lengths so that the LSTM ignores padded positions during its internal computations.

### 5.2.3 Output Layer

The BiLSTM outputs a hidden vector for every residue in the sequence.
A fully connected linear layer is then applied at each position to map this hidden vector to class scores.

By choosing the size of the output layer to match the number of classes, the same architecture can be used for both Q3 prediction (three classes), andQ8 prediction (eight classes).
The model produces one label prediction per residue.

### 5.2.4 Training Procedure

The BiLSTM model follows the same general training framework as the CNN:

- Loss function: Cross-entropy loss, with padding positions ignored
- Optimiser: Adam-based optimiser
- Learning rate control: Scheduler driven by validation performance
- Early stopping: Used to stop training when validation metrics stop improving
- Mixed precision: Used where available to speed up training

Because it can use information from both directions along the sequence, the BiLSTM serves as a strong sequence model baseline for both Q3 and Q8 secondary structure prediction.

### 5.2.5 Fine Tuning

The model performed badly from the beginning and hence was not explored further and was not fine tuned.

## 5.3 Transformer-Based Model

The Transformer model was used to learn both short-range and long-range relationships between amino acids. Unlike the BiLSTM, which reads the sequence from left to right (and right to left), the Transformer can look at all positions at the same time. This makes it useful for proteins, where residues that are far apart in the sequence can still influence each other’s structure.

#### 5.3.1 Input Representation

The model begins by converting each amino acid index into a learnable embedding vector.
Since Transformers do not automatically understand the order of positions in a sequence, we add positional encodings. These encodings give the model information about where each residue appears in the protein chain.

The final input to the Transformer is the sum of the amino acid embedding and its positional encoding.

#### 5.3.2 Transformer Encoder Layers

The core of the model is a stack of Transformer encoder layers.
Each layer has two main parts:

- Multi-head self-attention:
  This allows each residue to “look at” every other residue in the sequence.
  With multiple attention heads, the model learns different types of relationships at once.
- Feed-forward network:
  A small fully connected network that further processes the information learned during attention.

Layer normalisation and dropout are used to stabilise training.
A padding mask is also applied so that the Transformer ignores padded positions and only attends to real residues.

#### 5.3.3 Output Layer

After passing through the Transformer encoder, each residue has its own feature vector.
A final linear layer converts these features into class scores for either Q3 or Q8.
This means the model predicts one structure label per residue in the sequence.

#### 5.3.4 Training Procedure

The Transformer uses the same training setup as the other models:

- Cross-entropy loss with padding ignored
- Adam-based optimisation
- Learning-rate scheduling based on validation results
- Early stopping to prevent overfitting
- Mixed-precision training and gradient clipping for stability

Because it can attend to the whole sequence at once, the Transformer is able to learn patterns that stretch across long distances, making it a strong model for secondary structure prediction.

#### 5.3.5 Fine Tuning

The model performed badly from the beginning and hence was not explored further and was not fine tuned. The small dataset size is the reason for its bad performance. Other hybrid models perfomed much better

## 5.4 Hybrid CNN–BiLSTM–Transformer Model

The hybrid model combines a CNN block, a BiLSTM layer, and a Transformer encoder to leverage the strengths of all three architectures. The CNN focuses on local sequence motifs, the BiLSTM adds bidirectional context, and the Transformer models long-range relationships across the entire sequence.

### 5.4.1 Input Representation

The model uses the integer-encoded amino-acid sequences, which are first passed through an embedding layer ($d_{\text{model}} = 192$) with positional encodings added. This creates a position-aware representation of shape $(B, T, 192)$.

### 5.4.2 CNN Block (Local Features)

A small convolutional block extracts local patterns:

- Conv1d (192 → 128), ReLU, BatchNorm
- Conv1d (128 → 192), ReLU, BatchNorm

The sequence is transposed to match Conv1d format and then returned to $(B, T, C)$.
This block learns short-range motifs commonly found in helices and β-strands.

### 5.4.3 BiLSTM Layer (Context Features)

The CNN output feeds into a bidirectional LSTM:

- Input size: 192
- Hidden size: 256
- Bidirectional: True

This produces 512-dimensional per-residue features that capture information from both directions along the sequence. Packed sequences ensure padding is ignored.

### 5.4.4 Transformer Encoder (Long-Range Dependencies)

A Transformer encoder with 3 layers and 6 attention heads is applied to the BiLSTM output.
Feed-forward layers use dimension 768 and dropout.
A padding mask ensures attention is only computed over real residues.

This block allows each residue to attend to distant positions, capturing long-range dependencies the CNN and LSTM cannot fully model alone.

### 5.4.5 Output Layer

Finally, dropout is applied and a linear layer maps the Transformer output to class logits for Q3 or Q8.

### 5.4.5 Fine Tuning

The hybrid model was trained from scratch but used several tuning strategies to manage its higher capacity. The model was optimised with AdamW using a moderate learning rate (4.7e-03) and weight decay for regularisation. A OneCycleLR scheduler was applied across the full training run, allowing the learning rate to warm up and then gradually decay, which helped stabilise training for the transformer layers.

Label smoothing  and dropout were used to improve generalisation, and gradient clipping prevented unstable updates from the BiLSTM and attention layers. Training also used gradient accumulation to simulate larger batch sizes and mixed-precision training for efficiency. A simple early-stopping rule, based on validation accuracy, selected the best-performing model checkpoint.

These tuning choices helped the hybrid architecture learn effectively while avoiding overfitting or training instability.

## 5.5 ESM-Based Model

The ESM-based model uses pretrained protein language model embeddings instead of learning representations from scratch. Since ESM has been trained on millions of protein sequences, it provides rich per-residue features that already capture structural and evolutionary patterns. We then train our own classifier on top of these embeddings for Q3 and Q8 prediction.

### 5.5.1 Input Representation

ESM operates directly on the raw amino acid sequence, so no integer encoding is needed.
Each sequence from the train, validation, and test splits is passed into the pretrained ESM2 model, which outputs a high-dimensional embedding vector for every residue.

These embeddings replace the need for an embedding layer and preserve the actual sequence lengths.

### 5.5.2 ESM Embedding Extraction

For each sequence, ESM produces a tensor of shape (sequence_length, embedding_dim).
Since sequences have different lengths, they are stored as a list and padded only when forming batches.

These embeddings act as the feature backbone for the classifier.

### 5.5.3 Classifier Architecture

To make predictions from the ESM embeddings, we train a lightweight classifier that includes:

- A BiLSTM to learn sequential structure from the embeddings
- Optional convolution or attention layers to enhance features
- A final linear layer for Q3 or Q8 classification

Only the last layer changes between Q3 and Q8 tasks.

### 5.5.4 Training Procedure

Training follows the same setup as the other models:

- Cross-entropy loss with padding ignored
- AdamW optimisation and learning-rate scheduling
- Early stopping based on validation accuracy
- Mixed-precision training for efficiency

Because ESM already encodes meaningful information, the classifier trains faster and usually converges more reliably than models trained purely on token embeddings.

### 5.5.5 Fine Tuning

For the ESM-based approach, the pretrained ESM2 model is used only to generate fixed embeddings and is kept frozen during training. Fine-tuning is applied to the ESMProteinStructurePredictor classifier trained on top of these embeddings, separately for Q3 and Q8.

The classifier is trained with Adam using a small learning rate (4.1e-04) and up to 30 epochs. A ReduceLROnPlateau scheduler lowers the learning rate when the validation loss stops improving, while an early-stopping patience of 5 epochs prevents overfitting. Cross-entropy loss is used with the appropriate padding index (Q3_PAD_INDEX or Q8_PAD_INDEX) so padded positions are ignored.

Training uses mixed precision with GradScaler to keep updates stable and efficient on GPU. For both Q3 and Q8, the model state with the lowest validation loss is saved and later reloaded for final test evaluation.

These choices ensure that only the lightweight classifier head is fine-tuned, while still taking advantage of the rich, fixed representations provided by the ESM2 embeddings.

## 6. Experiments & Results

This section summarises the experiments conducted for Q3 and Q8 secondary structure prediction across all the models: CNN, BiLSTM, Transformer, Hybrid, and ESM-based classifier. Each model was trained using the same train/validation/test splits to ensure consistent and fair comparison.

### 6.1 Experimental Setup

All models were trained on the padded sequence tensors (or ESM embeddings) described earlier.
The training setup was consistent across models:

- Loss function: Cross-entropy with padding positions ignored
- Optimiser: Adam or AdamW depending on the model
- Batch size: 64
- Learning rate: Tuned per model, typically between 1e-4 and 5e-3
- Mixed-precision training: Enabled for speed
- Early stopping: Based on validation loss or validation accuracy
- Hardware used: GPU: CUDA

Both Q3 and Q8 tasks were trained separately for each model, with only the output layer dimension changed.

### 6.2 Evaluation Metrics

Model performance was assessed using:

- Overall per-residue accuracy
- Validation loss
- Test accuracy (primary comparison metric)

All metrics were calculated after removing padded positions.

### 6.3 Q3 Results

#### 6.3.1 CNN Performance

- Validation Accuracy: 70.70%
- Test Accuracy: 71%

#### 6.3.2 BiLSTM Performance

- Validation Accuracy: 59.31%
- Test Accuracy: 59.62%

#### 6.3.3 Transformer Performance

- Validation Accuracy: 50.04%
- Test Accuracy: NA

#### 6.3.4 Hybrid Performance

- Validation Accuracy: 68.46%
- Test Accuracy: 69.60%

#### 6.3.5 ESM-Based Performance

- Validation Accuracy: 81.6%
- Test Accuracy: 82.04%

### 6.4 Q8 Results

#### 6.4.1 CNN Performance

- Validation Accuracy: 52.49%
- Test Accuracy: 55%

#### 6.4.2 BiLSTM Performance

- Validation Accuracy: 46.30%
- Test Accuracy: 46.87%

#### 6.4.3 Transformer Performance

- Validation Accuracy: 39.79%
- Test Accuracy: 39.79%

#### 6.4.4 Hybrid Performance

- Validation Accuracy: 57.69%
- Test Accuracy: 58%

#### 6.4.5 ESM-Based Model Performance

- Validation Accuracy: 70.30%
- Test Accuracy: 70.45%

### 6.5 Overall Comparison

| **Model** | **Q3 accuracy** | **Q8 accuracy** |
|---|---|---|
| CNN | 71% | 55% |
| BiLSTM | 59.62% | 46.87% |
| Transformer | NA | 39.79% |
| Hybrid | 69.60% | 58% |
| ESM | 82.04% | 70.45% |

## 7. Discussion

The results show clear differences in how each model handles secondary structure prediction. Among the baseline architectures, the CNN performed the strongest, likely because its convolutional filters are well suited for capturing short, local patterns that dominate many Q3 and Q8 regions. The BiLSTM struggled more, reflecting the difficulty of learning long-range patterns purely through sequential processing, especially for longer protein chains. The Transformer, despite its ability to model global relationships, underperformed in this setting—most likely due to the relatively small model size and limited training budget available.

The hybrid model offered a noticeable improvement over all standalone baselines. By combining CNN, BiLSTM, and Transformer components, it was able to extract local motifs, sequential context, and longer-range dependencies within the same framework. This balanced architecture helped it generalize more effectively.

The ESM-based model achieved the strongest performance overall. Since ESM embeddings already encode rich evolutionary and structural information learned from large protein databases, the classifier built on top of these embeddings began from a far more informative representation than models trained from scratch. This advantage was reflected in both Q3 and Q8 predictions.

Overall, the trend suggests that the more contextual or pretrained knowledge a model can incorporate, the better it performs on both secondary-structure tasks.

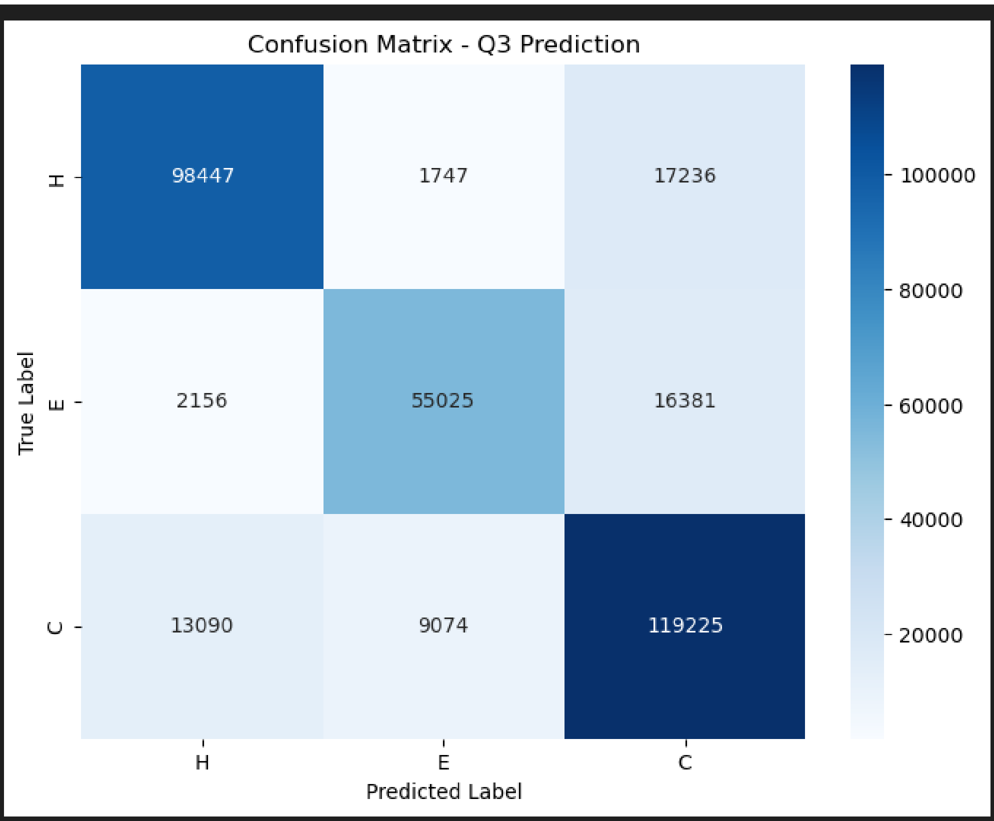
## 8. Conclusion

This project compared five deep-learning approaches to protein secondary structure prediction: CNN, BiLSTM, Transformer, a hybrid architecture, and an ESM-based model. Models trained purely from scratch were limited in how much structural information they could learn, especially for the more detailed Q8 classification task. The hybrid model improved on this by combining complementary feature extractors, showing the benefit of integrating multiple viewpoints into a single architecture.

The ESM model provided the strongest results, demonstrating the value of pretrained protein language models. These embeddings capture patterns that smaller models cannot learn from limited data alone, making them especially effective for residue-level prediction tasks. Overall, the findings highlight the importance of leveraging pretrained protein representations and suggest that future work should continue exploring models that integrate both structural priors and large-scale protein knowledge.

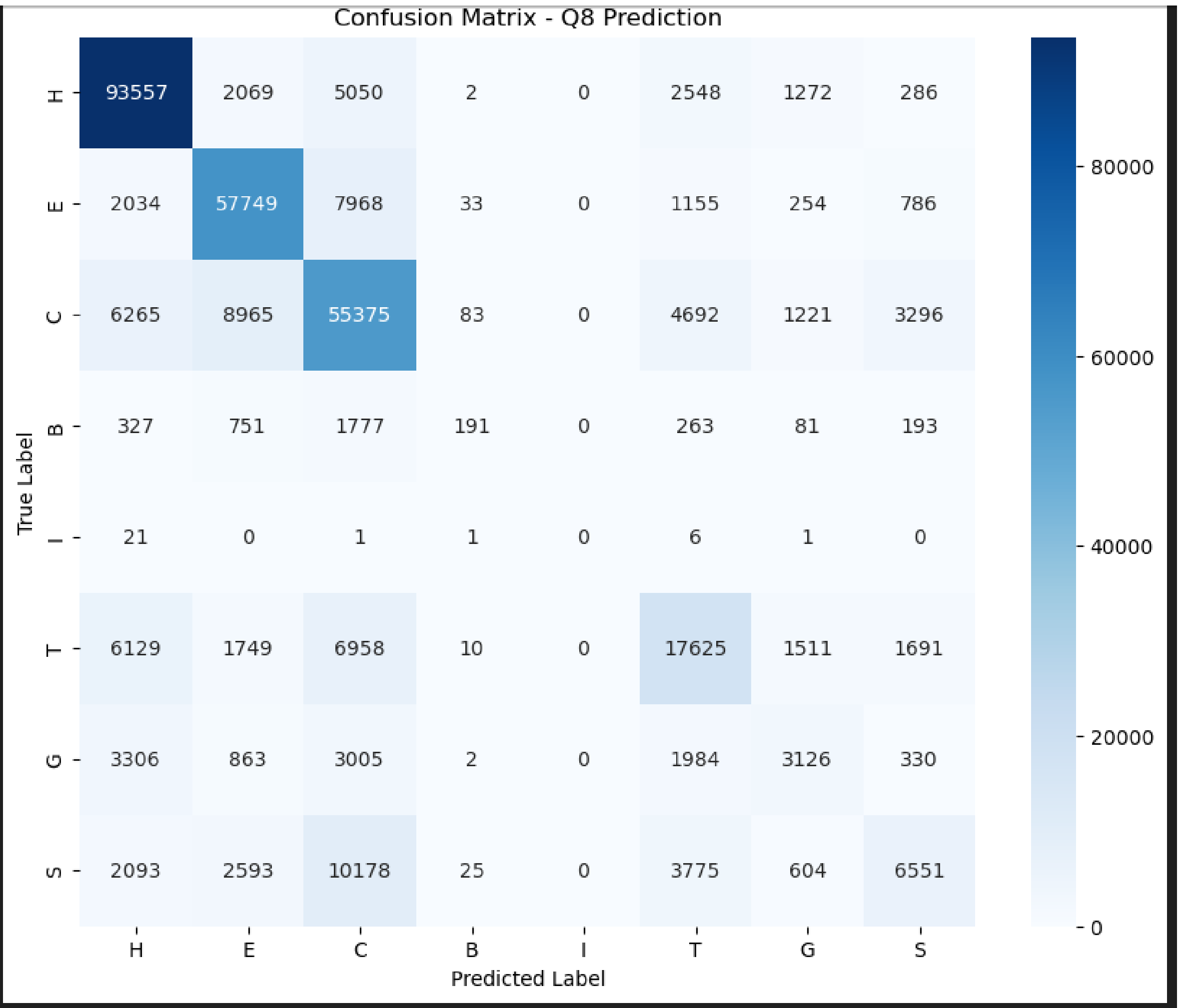
## 9. Appendix

### 9.1 Confusion Matrix for ESM Model (Q3)



```
✓ Confusion matrix saved to 'esm_confusion_matrix.png'
H (Helix): 0.8383 (83.83%)
E (Sheet): 0.7480 (74.80%)
C (Coil): 0.8432 (84.32%)
Q3 (Avg Per-Class): 0.8099 (80.99%)

=== Summary ===
Best Val Acc:  0.8174 (81.74%)
Test Acc:      0.8204 (82.04%)
Difference:    0.0030
```

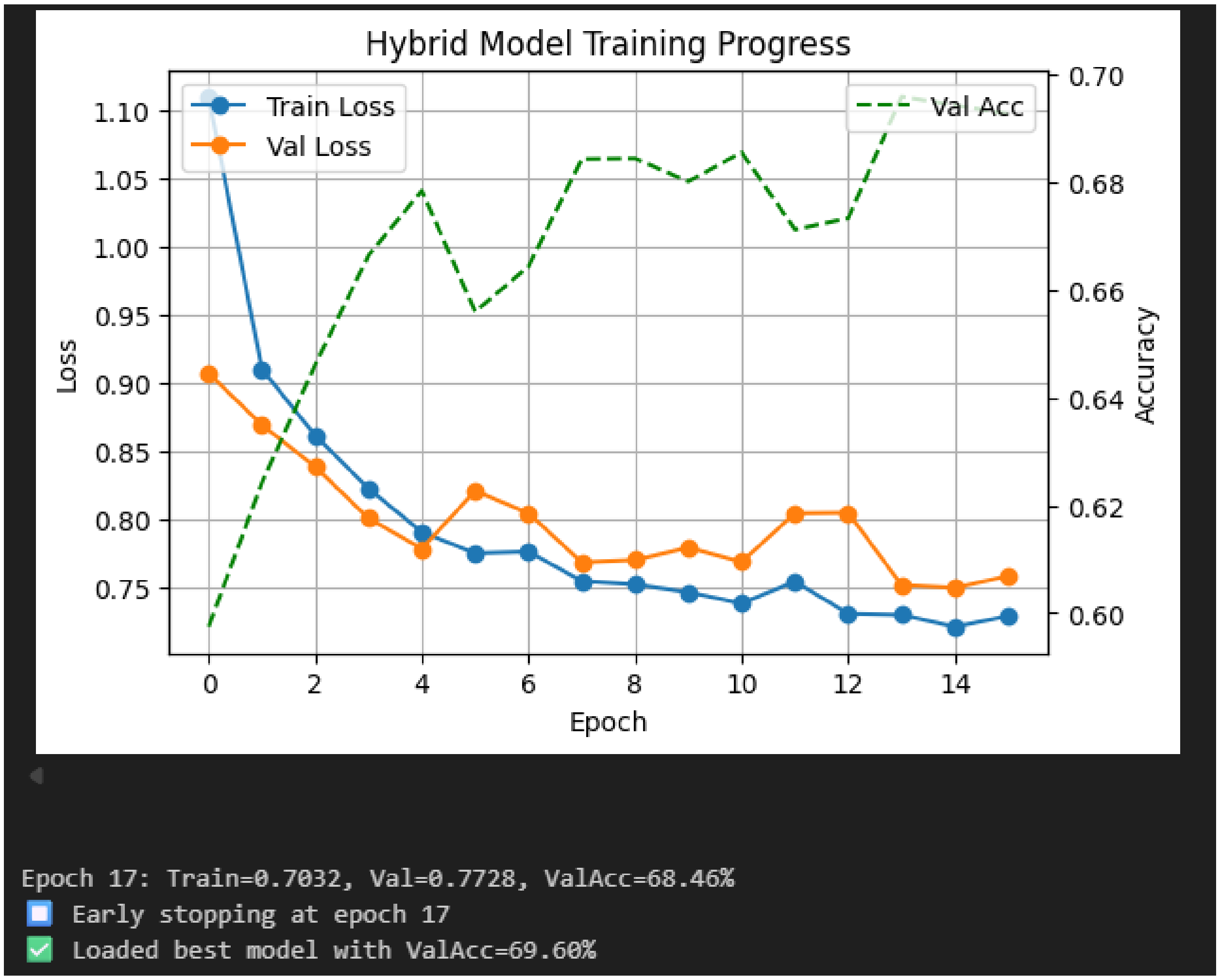
## 9.2 Confusion Matrix for ESM Model (Q8)



| | H | E | C | B | I | T | G | S |
|---|---|---|---|---|---|---|---|---|
| H | 93557 | 2069 | 5050 | 2 | 0 | 2548 | 1272 | 286 |
| E | 2034 | 57749 | 7968 | 33 | 0 | 1155 | 254 | 786 |
| C | 6265 | 8965 | 55375 | 83 | 0 | 4692 | 1221 | 3296 |
| B | 327 | 751 | 1777 | 191 | 0 | 263 | 81 | 193 |
| I | 21 | 0 | 1 | 1 | 0 | 6 | 1 | 0 |
| T | 6129 | 1749 | 6958 | 10 | 0 | 17625 | 1511 | 1691 |
| G | 3306 | 863 | 3005 | 2 | 0 | 1984 | 3126 | 330 |
| S | 2093 | 2593 | 10178 | 25 | 0 | 3775 | 604 | 6551 |

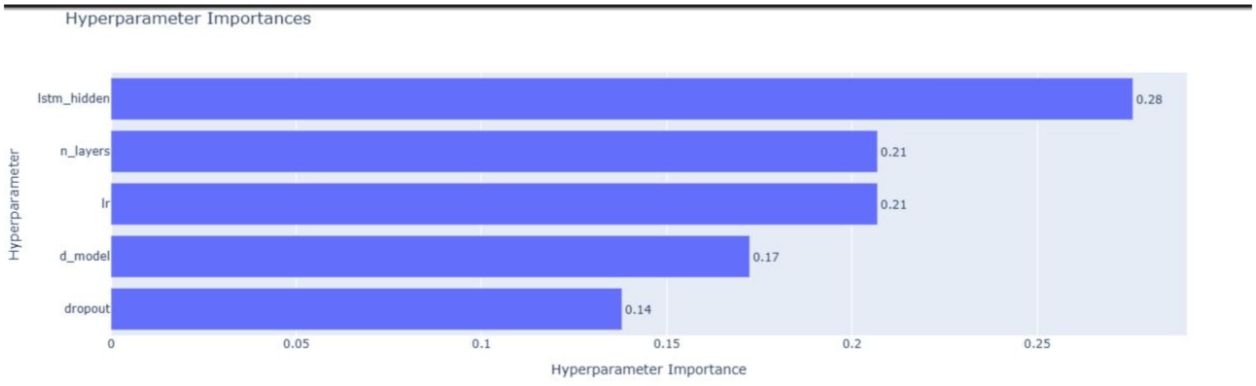
```
✓ Q8 Confusion matrix saved to 'esm_confusion_matrix_q8.png'

=== Q8 Summary ===
Best Val Acc:  0.7030 (70.30%)
Test Acc:      0.7045 (70.45%)
Difference:    0.0016
```

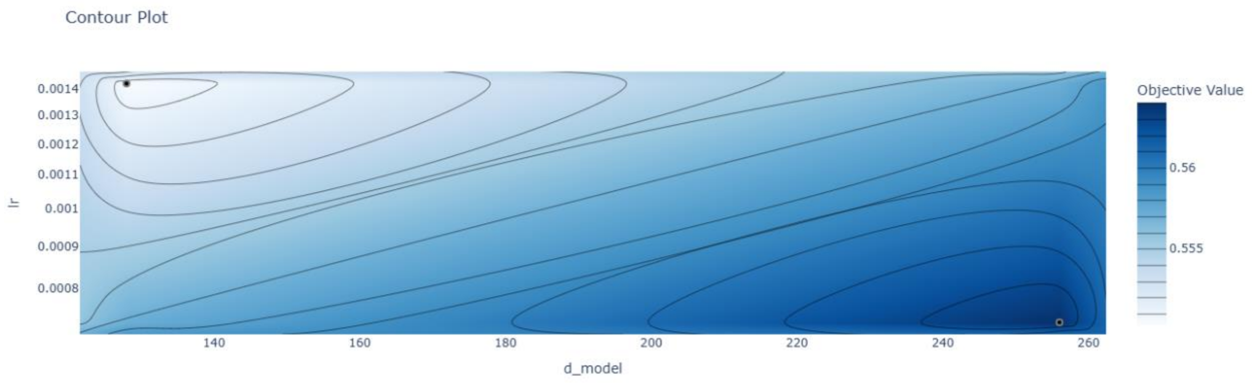
## 9.3 Hybrid model for Q3



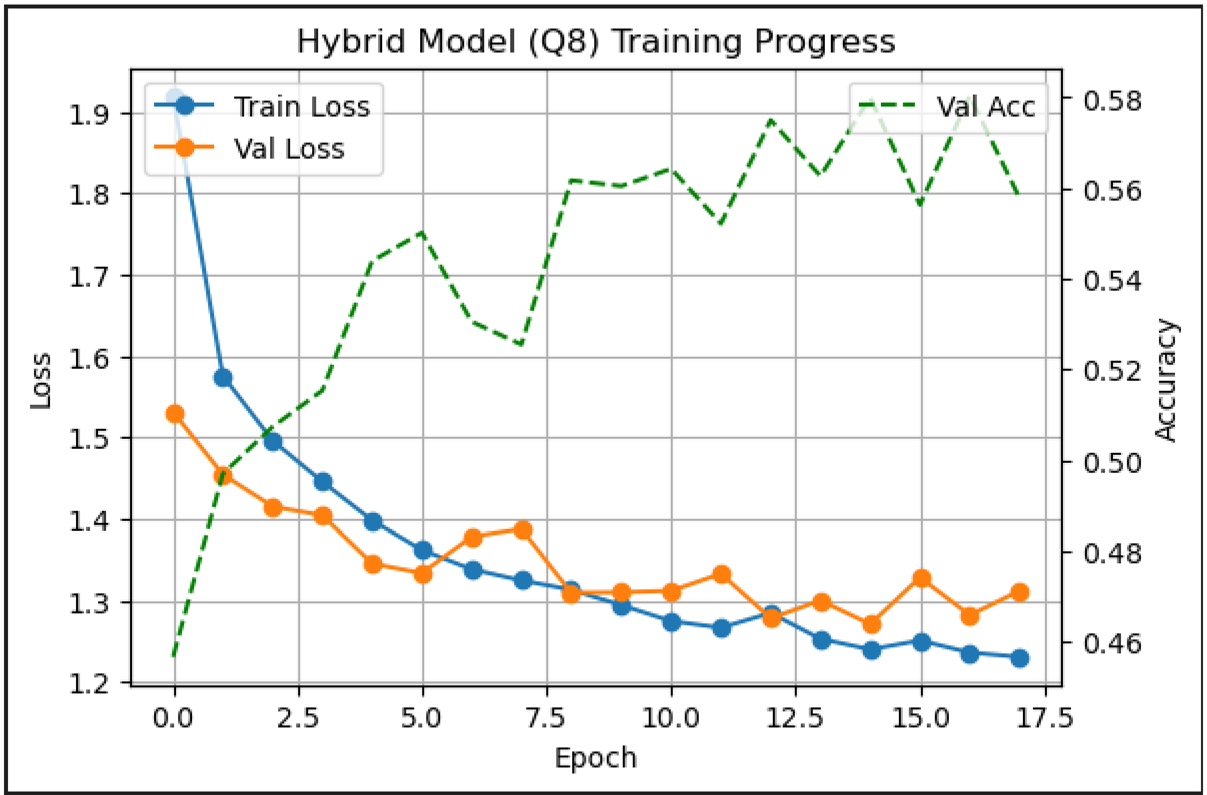
```
Epoch 17: Train=0.7032, Val=0.7728, ValAcc=68.46%
⏹ Early stopping at epoch 17
✅ Loaded best model with ValAcc=69.60%
```

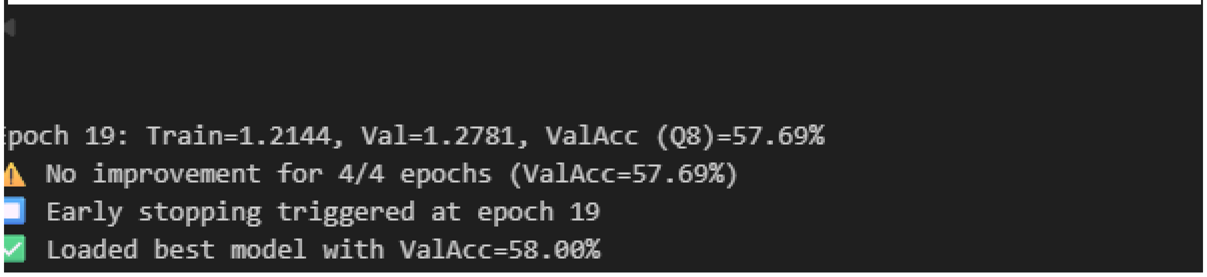
## 9.4 OPTUNA plot for Q8 hybrid





9.5 Hybrid Model for Q8





```
poch 19: Train=1.2144, Val=1.2781, ValAcc (Q8)=57.69%
⚠ No improvement for 4/4 epochs (ValAcc=57.69%)
  Early stopping triggered at epoch 19
✅ Loaded best model with ValAcc=58.00%
```

## 10. References

1. European Bioinformatics Institute (EBI). Introduction to Protein Structure: Secondary Structure. 2025. https://www.ebi.ac.uk/training/online/courses/introduction-protein-structure/secondary-structure/
2. Rives A. et al. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. PNAS, 118 (15): e2016239118.
3. Meta AI Research. ESM Protein Language Models (ESM2). https://github.com/facebookresearch/esm
4. Protein Data Bank (PDB). Repository of 3D Protein Structures. https://www.rcsb.org/