Protein Structure Prediction

Math 156: Machine Learning Final Project

Andrew Zhang, Aryan Dalal, Jiwoo Hyun, Steven Villanueva
University of California, Los Angeles (UCLA)

Fall 2025

Contents

1 Introduction

The goal of this project was to investigate how increasing the size of the ESM-2 protein inference model improved its accuracy in secondary structure prediction. To do this, we obtained embeddings from three versions of the ESM-2 model of increasing size and compared their performance when trained on a set of common classifiers. The results show that while increasing model size does consistently improve model accuracy, these improvements diminish as the model grows.

The secondary structure of a protein describes local patterns in the protein's physical structure, while a protein's primary structure is simply the linear sequence of amino acids that comprises it. With today's experimental methods, it is much simpler and cheaper to determine a protein's primary structure than its secondary structure. However, secondary structure is often far more indicative of a protein's function in the body. Being able to predict secondary structure from primary structure is thus a fundamental problem in computational biology.

The ESM-2 model attempts to solve this problem by treating it as a NLP problem, using modern transformer architectures to learn the "language" of protein sequences. When given a sequence of amino acids, ESM-2 outputs a vector embedding of each amino acid that captures the context of the other amino acids near it. These embeddings are what we used to classify secondary structure. We chose the 35M-, 150M-, and 650M-paramater models to see how a roughly fivefold increase in model size can improve performance. To compare these embeddings, we used logistic regression, linear SVC, and a two-hidden-layer MLP.

2 Data Overview

Our raw dataset consisted of 1,440,460 rows of amino acids taken from 6,148 unique protein sequences. Each row contained an amino acid's identity, the primary structure of the protein it was taken from, its position within its primary structure, and its secondary structure, which was our target variable. As all of our raw data was either discrete or categorical, no preprocessing was done before evaluating the data with the ESM-2 model.

Each version of ESM-2 that we used produced a uniquely sized embedding. The 650M-parameter model produced 1,280-dimensional embeddings for each amino acid, the 150M-parameter model produced 640-dimensional embeddings, and the 35M-parameter model produced 480-dimensional embeddings. These embeddings were used as our predictor variables in place of our raw data when performing our various classification techniques.

3 Methodology

In this project, we evaluate three models (Logistic Regression, Linear SVC, and a two-hidden-layer MLP) across three ESM embedding sizes (35M, 150M, and 650M). Our goal is to understand how the choice of model and the dimensionality of the embeddings jointly influence classification accuracy and class-balanced performance for protein secondary structure prediction.

We split the dataset into 70 percent training and 30 percent testing. During training, we applied feature scaling using standardization to prevent numerical instability and to ensure comparable magnitudes among input features. Hyperparameters for each model were selected using 3-fold cross-validation on the 150M embeddings. Because our primary objective was to study how performance varies with embedding dimensionality, we fixed the chosen hyperparameters and applied the same settings to the 35M, 150M, and 650M embeddings for a controlled comparison. All experiments were conducted in a GPU-accelerated environment.

For evaluation, we used accuracy, macro F1 score, and confusion matrices. The dataset exhibits strong class imbalance, so macro F1 was included to equally weight performance across all nine structural classes. Confusion matrices were used to visualize class-specific prediction patterns and to further illustrate the effects of imbalance on each model.

3.1 Logistic Regression

We use multinomial Logistic Regression (LR) as a linear baseline over the ESM embeddings to approach our classification problem. $L(y_1, \ldots, y_N; W) = \prod_{i=1}^{N} \prod_{j=1}^{k} p_{ij}^{\mathbf{1}(y_i=j)}$,

Our goal is to find:

$$\hat{W} = \arg\min_{W \in \mathbb{R}^{d \times k}} \left[\sum_{i=1}^{N} \log\Big(\sum_{q=1}^{k} \exp(\phi(x_i)^\top w_q) \Big) \; - \; \mathrm{tr}\Big(Y^\top \Phi^\top W \Big) \right].$$

To control overfitting we add L2 regularization,

$$\ell_{\mathrm{MLR}}(W) = \sum_{i=1}^{N} \log\Big(\sum_{q=1}^{k} \exp(\phi(x_i)^\top w_q) \Big) - \mathrm{tr}\Big(Y^\top \Phi^\top W \Big) + \frac{\lambda}{2} \|W\|_F^2, \qquad \lambda = \tfrac{1}{C}.$$

In terms of training, we standardize features (fit on train only) and optimize ℓ_{MLR} on GPU with `cuML`'s quasi-Newton solver. Because the dataset is imbalanced across $k = 9$ classes, we use class-balanced weights so that each row of Y contributes fairly where $Y \in \{0, 1\}^{N \times k}$ is the matrix with $Y_{ij} = 1(y_i) = j$.

Hyperparameters were selected on the 150M embeddings using a 3-fold CV metric over $C = \{0.1, 0.5, 1.0, 5.0, 10.0\}$; the best configuration was $C = 1.0$ with class balancing, which we then fixed and applied to the 35M, 150M, and 650M embeddings for a controlled

comparison, leading to the results below.

3.2 Linear SVC

The Linear Support Vector Classifier is a linear classifier that seeks a decision boundary maximizing the margin between classes. Conceptually, it is similar to the perceptron in that it learns a separating hyperplane, but differs by optimizing a margin-based hinge-loss objective rather than relying purely on misclassification errors. Because this project requires GPU-accelerated training, we use the cuML implementation of LinearSVC, which provides a scalable linear SVC suited for high-dimensional embeddings.

Hyperparameters were selected through 3-fold cross-validation on the 150M embeddings. We searched over

$$C \in \{0.5, 1, 2, 5, 10, 15\}, \qquad \texttt{max\_iter} \in \{2000, 3000, 5000, 8000\},$$

and obtained the best-performing configuration $C = 0.5$ and `max_iter` $= 2000$.

The Linear SVC optimizes the hinge-loss objective

$$\min_{w} \ \frac{1}{2}\|w\|^2 \ + \ C\sum_{i=1}^{n} \max(0,\, 1 - y_i\, w^\top x_i),$$

where the parameter C controls the trade-off between maximizing the margin and penalizing misclassified samples.

All input features were standardized using z-score normalization prior to training. We enabled `class_weight="balanced"` so that all classes receive equal treatment during training. All SVC models were trained using the GPU-accelerated `cuML` implementation of `LinearSVC`.

3.3 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a supervised learning model that learns a function $f : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ by training a dataset. The MLP is a type of a Feedforward Neural Network in the sense that information flows only in one direction. A MLP can be used for both regression or classification tasks and is in fact, quite similar to Logistic Regression; however, a neural network is able to incorporate multiple nonlinear layers during the training process (such as ReLU) allowing the model to be more robust.

The NVIDIA cuML library does not yet have the Multi-layer Perceptron model and as a result, we design our own model architecture using PyTorch. However, as we use cuPy arrays, much of the pre-processing involved converts cuPy GPU-accelerated arrays to numPy CPU arrays that can be processed by our model.

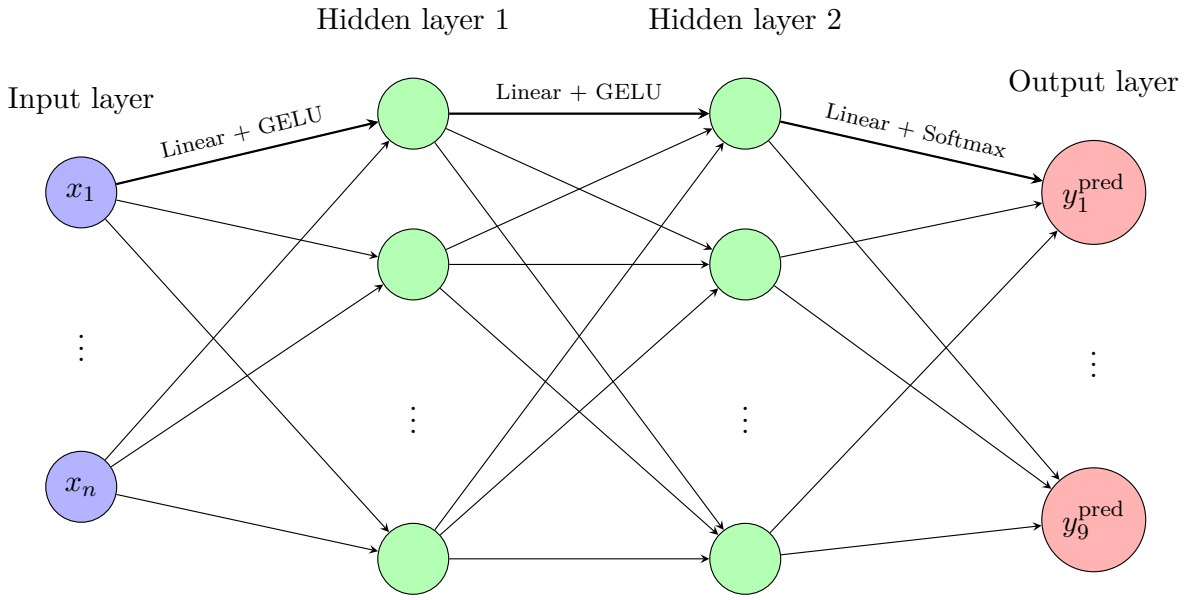


Figure 1: 3-Layer Multi-layer Perceptron Classifier

The 3-Layer Multi-Layer Perceptron was decided as the best model for this task after experimenting with a 2-Layer MLP and a 4-Layer MLP. We understood that there was significant improvement in performance with a 3-Layer MLP but not much of an increase if we were to choose 4-Layers. The Multi-layer Perceptron optimizes the Cross Entropy Loss function

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^{512}}{\arg\min} \left[\ell_{\text{MLP}} = \sum_{s=1}^{n} \ell\left(\mathbf{y}_s, \hat{\mathbf{y}}\left(\mathbf{x_s}; \mathbf{w}\right)\right) \right] = \underset{\mathbf{w}\in\mathbb{R}^{512}}{\arg\min} \left[-\sum_{i=1}^{n} \mathbf{1}\left(\mathbf{y}_s = \text{class } i\right) \log\left(\hat{y}_i\left(\mathbf{x}_s; \mathbf{w}\right)\right) \right]$$

We optimize this using Adaptive Moment Estimation (Adam) which is simply an extension of Stochastic Gradient Descent.

During experimentation, we made the decision to switch ReLU with GELU activation. GELU stands for Gaussian Error Linear Unit which is smooth (i.e., it's differentiable everywhere) and non-monotonic. ReLU is differentiable everywhere except at $x = 0$. What makes GELU a better fit for the activation choice here is that given our minority classes, GELU is able to capture these patterns much more efficiently compared ReLU empirically.

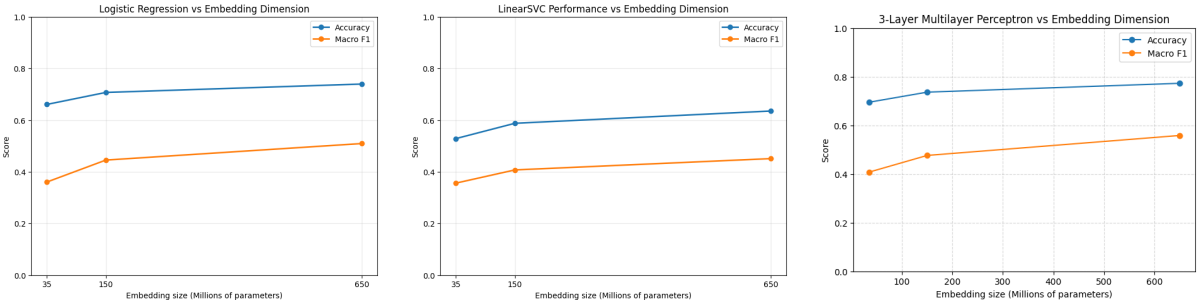
For every embedding dimension (650, 150, 35), the training loop for the MLP defines a dropout for regularization as well as a learning rate scheduler based on a validation split. We do so because a neural network may be prone to *memorizing* rather than *learning* and a validation loss allows us to verify that if the training loss is decreasing but the validation loss remains consistent, the model will terminate the training process and pick the best epoch (by validation accuracy) instead of training till the end.

4 Results and Discussion

We evaluated three supervised learning models on ESM protein embeddings of different sizes (35M, 150M, and 650M parameters) to understand how both model choice and embedding dimension influence secondary structure prediction performance. The models were Logistic Regression with class balancing, Linear Support Vector Classification, and a three-layer MLP with hidden dimensions 512 and 256. All models were trained and tested under identical splits to ensure a fair comparison.

Across all three models, both accuracy and macro-F1 increase consistently as the embedding size grows. Larger embedding models provide richer representations, allowing even simple linear classifiers to capture more structural information. However, the increase is not uniform across models. Some classifiers benefit much more from embedding quality than others.

Before examining the detailed results, Figure 2 below summarizes how accuracy and macro-F1 evolve with embedding size for the three classifiers.



(a) Logistic Regression (b) LinearSVC (c) MLP

Figure 2: Performance across embedding sizes for all models.

Figure 2 illustrates that all models benefit from larger embedding sizes, with accuracy and macro-F1 rising smoothly from 35M to 650M. Logistic Regression and the MLP both begin with relatively strong performance at 35M and maintain consistent improvements as embedding size increases. LinearSVC shows a similar upward trend, although the magnitude of these improvements is not visually obvious from the plot alone.

The numerical results in Table 1 reveal that LinearSVC experiences the largest absolute improvement as embedding size increases. From 35M to 650M embeddings, both accuracy and macro-F1 for LinearSVC rise more sharply than for the other two classifiers. Logistic Regression and the MLP, on the other hand, show more moderate gains because they already start at comparatively high performance levels at 35M. While the MLP achieves the best overall scores at each embedding dimension, the linear models exhibit strong and meaningful improvements driven primarily by the quality of the pretrained embeddings.

Notably, Logistic Regression delivers competitive performance with minimal compu-

Table 1: Performance comparison across embedding sizes for all models

Model	Embedding Size	Accuracy	Macro F1
Logistic Regression	35M	0.6608	0.3606
	150M	0.7071	0.4454
	650M	0.7396	0.5094
LinearSVC	35M	0.5286	0.3563
	150M	0.5877	0.4073
	650M	0.6352	0.4511
3-Layer MLP (512, 256)	35M	0.6951	0.4078
	150M	0.7363	0.4764
	650M	0.7729	0.5584

tational cost, making it an efficient baseline. The MLP outperforms the linear models overall, but its computational requirements are significantly larger. These differences highlight a practical tradeoff between model complexity and resource constraints.

The confusion matrices in Figures 3–5 further support these observations. All models consistently identify the dominant secondary-structure classes, while performance on rare classes varies. Logistic Regression exhibits increasingly sharper diagonal patterns as embedding size grows. LinearSVC shows the largest absolute improvement when the embedding size increases, but its confusion matrices remain comparatively diffuse and low-contrast due to its overall weaker performance. As a result, the gains are less visually apparent despite being the largest numerically, a pattern that aligns with the absolute differences reported in Table 1. The MLP achieves the most balanced class-wise performance overall, although its higher confidence results in slightly noisier predictions for ambiguous sequences.

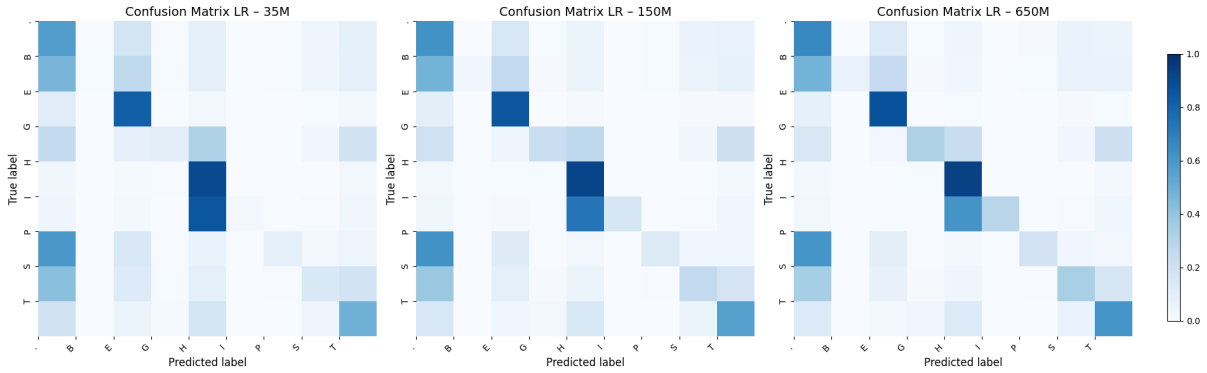


Figure 3: Confusion matrices for Logistic Regression across embedding sizes.

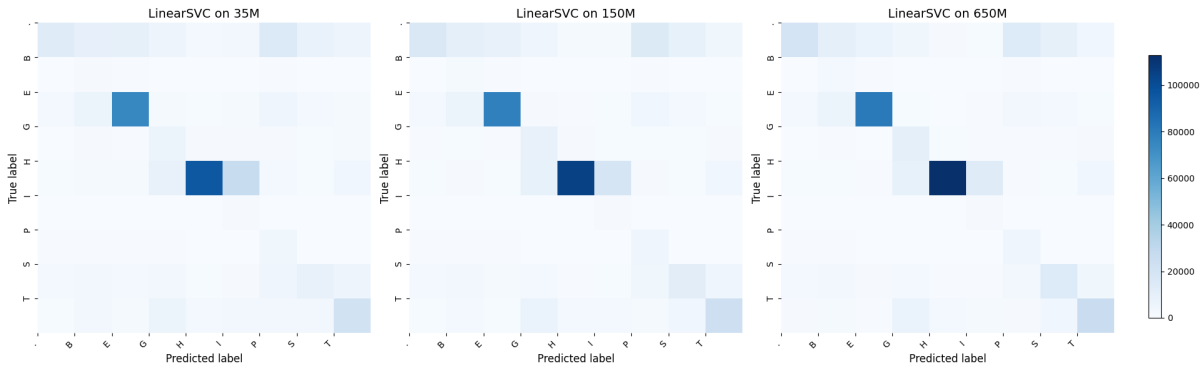


Figure 4: Confusion matrices for LinearSVC across embedding sizes.

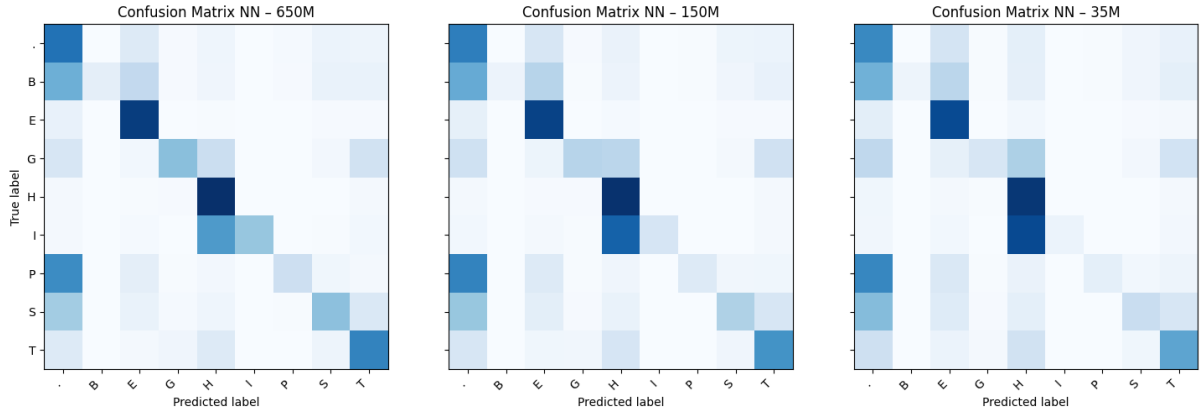


Figure 5: Confusion matrices for MLP across embedding sizes.

Overall, embedding quality has a substantial impact on performance regardless of the classifier used. The MLP yields the strongest absolute results at each embedding size, while LinearSVC shows the largest improvement with increasing embedding dimension. Logistic Regression offers a compelling balance between performance and computational efficiency. Together, these findings suggest that pretrained embedding quality plays a dominant role, and model choice allows practitioners to balance accuracy and computational cost according to application needs.

5 Conclusion

In this study, we evaluated how the size of ESM-2 embeddings influences downstream secondary-structure prediction across three classifiers: Logistic Regression, LinearSVC, and a 1-hidden-layer MLP. Due to the nature of the 9-way imbalanced classification problem, we emphasized Macro-F1 alongside accuracy.

We found that performance improves monotonically with embedding size for all models. Relative to the Macro-F1 chance level which is $\approx 1/9$, the 650M embeddings yield large gains for every classifier. The MLP achieves the highest absolute scores (up to ≈ 0.773 accuracy and 0.558 Macro-F1), while Logistic Regression provides a strong, stable linear baseline (up to ≈ 0.740 accuracy and 0.509 Macro-F1) with much lower complexity and cost. LinearSVC shows the same trend but is, interestingly, slightly behind Logistic Regression in both accuracy and Macro-F1. Improvements are substantial from 35M to 150M and noticeably smaller from 150M to 650M, indicating diminishing returns as model size grows.

Some possible improvements are:

(i) Instead of fixing hyperparameters tuned on 150M, we could do per-size retuning and it could close small gaps.

(ii) More targeted imbalance handling (e.g., calibrated class weights, focal loss, or cost-sensitive thresholds) may increase Macro-F1 for rare classes.

Overall, the results support a clear conclusion: embedding quality dominates model choice in this task. Larger ESM-2 embeddings consistently improve balanced multi-class performance, with the 650M variant delivering the strongest outcomes. Interestingly, the well-regularized Logistic Regression performed best with cost/benefit consideration.

6 References

The dataset used in this project is from UCLA's COM SCI 121 course project developed by Professor Eleazar Eskin. That project involved using any publicly available methods for protein structure prediction, but the ESM-2 model was specifically mentioned as a prominent method. All of our raw data came from this course dataset.

The ESM-2 models were released by Meta alongside their "Evolutionary-scale prediction of atomic level protein structure with a language model" paper. They were accessed through the HuggingFace transformers package. The specific ESM-2 models used in this project were esm2_t33_650M_UR50D, esm2_t30_150M_UR50D, and esm2_t12_35M_UR50D.

Both getting the ESM-2 embeddings and building the FFNN were done with PyTorch, using CUDA on Google Colab's A100 GPU for more efficient computation. We used cuML packages to run LinearSVC and LogisticRegression on the GPU as well. For calculating our error metrics, we used scikit-learn.metrics. Our visualizations were done using Matplotlib's pyplot and Seaborn.

7 Credits

Andrew: Logistic Regression experiments, methodology writing, conclusion writing
Aryan: MLP experiments, methodology writing, slide preparation
Jiwoo: Linear SVC experiments, methodology writing, results and discussion writing
Steven: Dataset setup, introduction writing, data overview writing