Benchmarking Transformer-Based Embeddings for Protein Subcellular Localization: A Comparative Study of LSTMs and Protein Language Models

Melodie Li[1,*]

[1]Computer Science Department, University College London, Gower Street, WC1E 6BT, London, United Kingdom

Abstract

In this study, we present a systematic comparison of computational models for protein subcellular localization (PSL) prediction across four compartments—cytosolic, secreted, mitochondrial, and nuclear—using a dataset of 7,458 training sequences and 20 test sequences. Our modeling framework spans classical machine learning baseline (SVM), various LSTM-based deep learning architectures, and transformer-based embedding methods with MLP classifiers. Results demonstrate that using embeddings generated by protein language models, particularly **ESM2-T33**, outperform other methods by achieving **83.4% macro-F1** and **81.9% accuracy** on the validation set. Overall, our findings highlight the efficacy of large pretrained contextual embeddings for PSL prediction and lay the groundwork for more interpretable and generalizable models.

1 Introduction

Predicting protein subcellular localization (PSL) is essential for understanding protein function in eukaryotic cells, as many proteins must be delivered to specific compartments—such as the cytosol, mitochondria, nucleus, or secretory pathway—to fulfill their biological roles. Mislocalization of proteins can disrupt cellular processes and contribute to numerous pathologies, including neurodegenerative diseases and cancer [1, 2]. Although experimental approaches of PSL via fluorescence microscopy or mass spectrometry remain highly accurate, they are expensive and time-consuming [3]. Consequently, computational PSL prediction has become indispensable for preliminary PSL prediction and guiding experimental validation.

Problem Statement

This work address the supervised classification of eukaryotic proteins into four major subcellular locations: cytosolic, secreted/extracellular, mitochondrail, and nuclear. Our goal is to develop robust predictive models that can generalize across diverse protein lengths and compositions, thereby supporting downstream applications in functional genomics and drug target discovery. Despite ongoing advancements in machine learning, precise discrimination among these subcellular compartments, particularly the cytosolilc-nuclear boundary, remains challenging due to overlapping sequence features, complex motif patterns, and limited large-scale annotated data [4, 5] .

Existing Approaches and Gaps

Early PSL predictors typically relied on handcrafted features (e.g. amino acid composition, signal motifs, and physiochemical properties) coupled with classifiers such as Support Vector Machines [6]. While these methods offered interpretable feature importance, they often struggled with the intricate dependencies inherent in protein sequences (e.g. long-range interactions), leading to suboptimal performance when motif distributions overlap, and were highly dependent on domain-specific feature engineering.

Deep learning revolutionized PSL prediction by learning feature representations directly from protein sequences. Convolutional Neural Network (CNNs) excel at detecting local motifs [7], whereas Recurrent Neural Networks (RNNs), including LSTM-based architetures, capture long-range dependencies [8, 9].

More recently, Transformer-based protein language models, such as ProtBERT [10, 11] and Evolutionary Scale Modeling (ESM) [12], have demonstrated state-of-the-art performance in multiple PSL tasks by treating protein sequences like natural language. These models leverage attention mechanisms to encode complex context dependencies across entire protein chains. Furthermore, graph neural networks (GNNs) [13, 14] have started to integrate protein-protein interaction data, offering a complementary perspective on subcellular localization but often increasing model complexity and cost.

Despite these advances, challenges remain. Many DL-based approaches require large, high-quality annotated sequence data, which are not always available for specialized subcellular compartments. Additionally, proteins with multiple localizations (e.g. dynamic trafficking between

cytosol and nucleus) are difficult to categorize under standard single-label frameworks [15]. Computational cost and interpretability also limit widespread adoption; while some methods leverage attention weights, integrated gradients or saliency maps to identify influential residues [9, 6], many biologists often require mechanistic explanations behind model predictions, highlighting a persistent interpretability gap.

Objectives and Paper Overview

To address these gaps, this study presents a systematic comparison of three major modeling paradigms for PSL prediction:

- **Classical ML Baseline.** We employ a Support Vector Machine (SVM) classifier trained on domain-specific (handcrafted) features to establish a baseline for computational efficiency and accuracy, building on its documented success in early PSL methods[16, 17].
- **LSTM-based Deep Learning.** We explore LSTM variants (e.g., BiLSTM, LSTM + attention, CNN–LSTM hybrids) to capture long-range dependencies in protein sequences. These architectures learn task-specific representations without heavy reliance on manually crafted features.
- **Transformer-Based Embedding Methods.** We apply pretrained protein transformers such as ESM2 and ProtBERT for contextual embedding generation, followed by a shallow MLP classifier. By leveraging large-scale protein language models, we aim to outperform conventional RNN-based solutions.

2 Data

Training Data The training set consists of 7,458 eukaryotic protein sequences, distributed across the four subcellular locations as follows:

- Cytosolic Proteins (2,463)
- Secreted/Extracellular Proteins (1,236)
- Mitochondrial Proteins (1,023)
- Nuclear Proteins (2,002)

It exhibits a moderately imbalanced distribution, with nuclear and cytosolic proteins being the most prevalent, while the other two are relatively underrepresented, necessitating class-balancing techniques such as during training.

Test Data The test set comprises 20 protein sequences without labels, limiting the possibility of quantitative performance metric son this subset. Instead, these data server as a final blind set for testing the model's generalizability, although its small size raises concerns about whether it fully reflects the diversity of eukaryotic proteins observed in the training set.

Feature Extraction

To leverage the information embedded in each protein sequence, we used a two-pronged feature strategy: (1) **handcrafted features** rooted in biochemical knowledge, and (2) **transformer-based embeddings** that offer high-dimensional contextual representations.

Handcrafted Features

1. **Amino Acid Composition** Each protein sequence was converted into a 20-dimension vector specifying the relative frequency of each canonical residue $(A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y)$. This broad chemical profile often reflects functional domains or subcellular targeting cues.
2. **Sequence Length** Given the marked variability in our protein lengths (see EDA, Section 2.4), length is included as a simple numeric features. Longer proteins may contain multiple domains or signal peptide relevant to subcellular targeting.
3. **Molecular Weight** Computed by summing the average residue masses for each amino acid in a sequence. High molecular weight can correlate with multi-domain architectures, which may localize to certain organelles.
4. **Isoelectric Point (pl)** Estimated using standard side-chain dissociation constants. Proteins with extreme pl values may favor compartments with compatible pH environments such as mitochondria.
5. **Signal Motifs** We flagged known targeting motifs (e.g., nuclear localization signals, mitochondrial targeting peptides, ER-retention sequences). The presence/absence of such motifs can strongly indicate subcellular location.
6. **Hydrophobicity & Net Charge**
   a. **Hydrophobicity**: Average Kyte–Doolittle index. Proteins with high hydrophobic regions often associate with membranes (e.g., secreted or organelle-bound).
   b. **Net Charge**: Calculated from positive and negative residues at physiological pH (7.0). Highly charged proteins can favor nuclear or cytoplasmic environments.

Embedding-Based Features

We also leveraged transformer-based embeddings from large-scale protein language models from ProtTrans [18](e.g., ProtBERT, ProtAlbert, ProtXLNet) and ESM [12] (e.g., ESM2-T6, ESM2-T33) framework, capturing richer, context-dependent residue relationships:

- **ProtBERT** [10] ProtBERT applys the BERT (Bidirectional Encoder Representations from Transformers) paradigm to learn deep, contextual embeddings from billions of amino acids across diverse organisms. By predicting masked tokens in a bidirectional fashion, it captures both local motifs and global conformational cues, making it suitable for tasks requiring balanced attention to short and long-range dependencies.
- **ProtAlbert** [19] ProtAlbert is a lightweight variant of the ProtBERT family, leveraging the ALBERT factorized embedding parameterization to reduce model size without sacrificing too much representational capacity.
- **ProtXLNet** [20] ProtXLNet adpots the XLNet framework, employing a permutation-based language modeling objective rather than a strictly masked approach. This enables the model to use bidirectional context while retaining the flexibility of an auto-regressive mechanism. As a result, ProtXLNet can discover nuanced sequence

relationships that might be missed by purely masked language models.

- **ESM2-T6 and ESM2-T33** Developed by [12], ESM2 represents the next generation of Evolutionary Scale Modeling approaches, refining the original ESM architecture with improved training protocols. The T6 variant contains 6 layers and 8M parameters, favoring speed and efficiency for moderate-scale tasks, while the T33 variant scales up to 33 layers and 650M parameters, offering deeper contextual representations of protein sequences at the cost of higher computational demand. Both ESM2-T6 and ESM2-T33 incorporate evolutionary information (e.g., multiple sequence alignments) and masked language modeling.

These embeddings are aggregated into vector representation for each residue (token), either by [CLS] token or mean pooling, then used as high-level input features for downstream classification.

Exploratory Data Analysis (EDA)

A preliminary EDA evaluated class distributions, feature correlations, and motif frequencies to guide model design and highlight potential classification hurdles.

Sequence Length Distribution Figure 1 show histograms

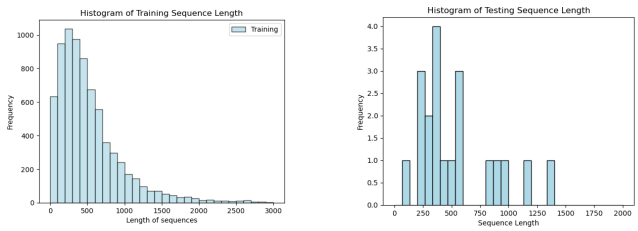


Fig. 1. Histogram of Training (Left) and Test (Right) Sequence Length Distribution.

of sequence lengths. The training set is right-skewed: most proteins are under 2,000 residues, peaking around 200 – 400 amino acids, but with a modest tail extending to 3,000 residues. This skewness is typical of eukaryotic protein, where a majority of proteins cluster at relatively moderate lengths, but certain multi-domain proteins inflate. The test set of just 20 sequences shows a less smooth distribution, typically under 1,000 residues. This discrepancy could pose a distribution mismatch, as longer sequences in the training data might influence feature importance and model focus.

Feature Correlation Analysis Figure 2 presents a correlation heatmap for the handcrafted features:

1. **Sequence Length** correlates strongly with **Molecular Weight**, reflecting the simple linear relationship between the number of residues and total mass.
2. Certain **amino acid composition** features have moderate negative correlations; for example, an increased fraction of one residue (e.g., leucine) often implies a decreased fraction of another (e.g.,glycine).

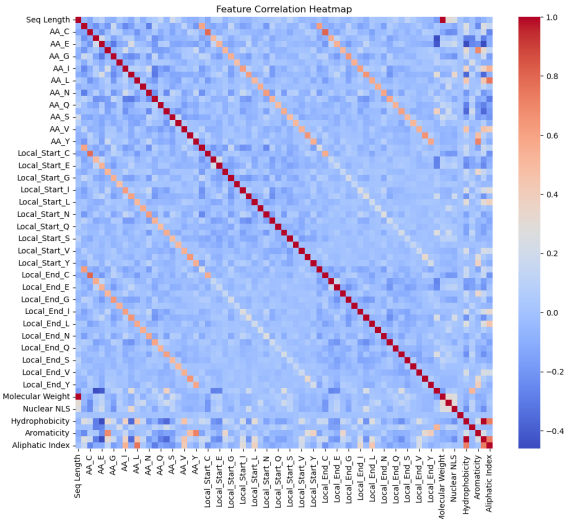


Fig. 2. Feature Correlation Heatmap.

3. **Motif indicators** such as nuclear localization signal exhibit lower correlations overall but can co-occur with net charge in nuclear proteins.

Isoelectric Point (pI) by Class Looking at the box plots of isoelectric point values in Figure 3 across each protein class, we observed that Mitochondrial proteins often have higher pI values (median ~9), aligning with the observation that many mitochondrial transit peptides are positively charged to facilitate import via electrochemical gradients. Cytosolic proteins tend to have a lower median pI (~ 6), reflecting a balance of acidic and basic residues suited to the relatively neutral cytosolic environment. Consequently, isoelectric point remains a key distriminator for certain compartments.

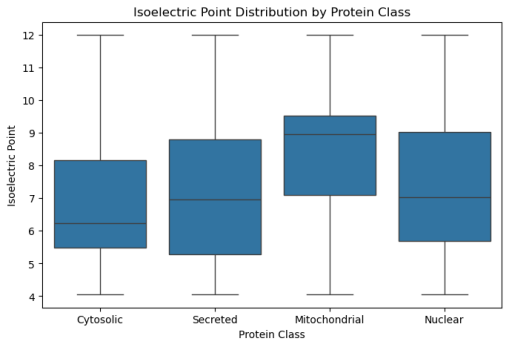


Fig. 3. Isoelectric Point Distribution by Protein Class.

Motif Presence Across Classes Figure 4 shows the occurrence of key subcellular motifs in the training data. Notably, a large fraction of protein appear to carry either a mitochondrial targeting motif or a nuclear localization signal (NLS), while endoplasmic reticulum (ER) retention motifs are extremely rare or absent in our training data. Several proteins lack any recognized motif, implying they may reply on less-characterized or non-canonical signals (e.g., post-translational modifications) for localization. This distribution underscores

the heterogeneity of subcellular trafficking mechanisms and the classification challenges if motif-based signals alone are insufficiently robust.

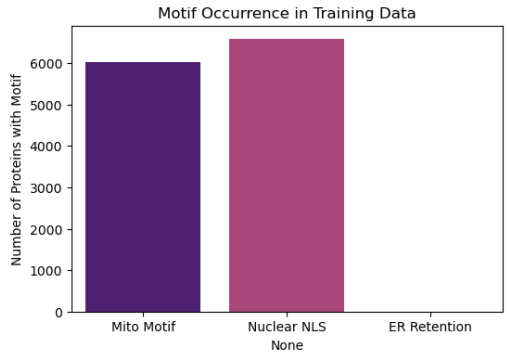


Fig. 4. Motif Occurence in Training Data Across Classes.

3 Methods

This section details the computational models used to predict protein subcellular localization. We begin with a classical machine learning baseline (Support Vector Machine), progress to LSTM-based deep learning architectures, and conclude with transformer-based embedding approaches (ProtBERT, ProtXLNet, ProtAlbert, ESM2) coupled with multi-layer perceptron (MLP) classifiers.

Classical Machine Learning Models

Support Vector Machine [21] Our multi-class SVM extends the binary SVM formulation to four subcellular classes using a one-vs-rest (or one-vs-one) scheme. In the linear case, a binary SVM seeks a hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ that maximizes the margin between positive and negative examples:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \quad \text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,$$

where C balances margin width against classification errors ξ_i. Non-linear kernels (e.g., RBF) project data into a higher-dimensional feature space for improved class separability. Handcrafted features (Section 2.3.1) were used as input to the SVM baseline.

Deep Learning Models

Proteins are inherently sequential, prompting the use of recurrent neural networks (RNNs) designed to capture positional and contextual features over long ranges. We experimented with several LSTM-based architectures and hybrids incorporating attention or CNN layers.

LSTM Architecture[22] Long Short-Term Memory (LSTM) networks mitigate the vanishing/exploding gradient issues encountered in standard RNNs. Figure 5 is the fundamental building block. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ be the input embeddings for a protein sequence of length T. The LSTM updates are:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i\right), \\
\mathbf{f}_t &= \sigma\left(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f\right), \\
\mathbf{o}_t &= \sigma\left(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o\right), \\
\tilde{\mathbf{c}}_t &= \tanh\left(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}$$

where $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ represent input, forget, and output gates, respectively; $\mathbf{c}_t$ is the cell state; $\mathbf{h}_t$ is the hidden state; σ is the sigmoid function; and $\odot$ denotes element-wise multiplication. The forget gate $\mathbf{f}_t$ modulates how much of the previous cell state $\mathbf{c}_{t-1}$ to retain, while $\mathbf{i}_t$ controls how much new information to add from $\tilde{\mathbf{c}}_t$.

Output Layer: After the final time step ($t = T$), we typically apply a dense layer on $\mathbf{h}_T$ (or a pooled version of $\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$) to produce four logits, then apply softmax to classify the protein into one of the subcellular classes.

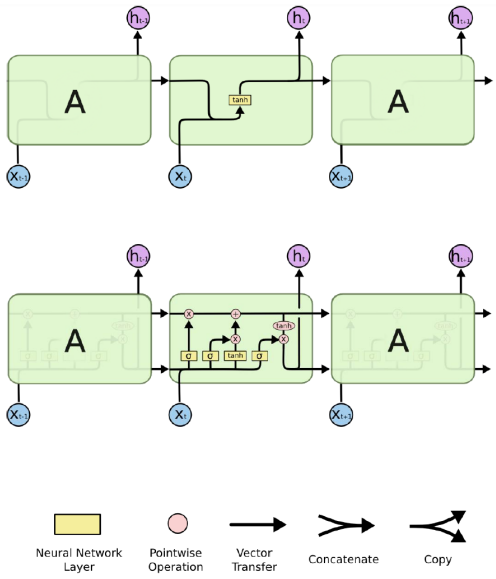


Fig. 5. Vanilla Long Short-Term Memory (LSTM) Model Architecture.

Bidirectional LSTM (BiLSTM) A BiLSTM, as depicted in 6 extends the LSTM architecture by processing the sequence in both forward ($t = 1 \rightarrow T$) and backward ($t = T \rightarrow 1$) directions and concatenates the hidden states:

$$\mathbf{h}_t^{(f)} = \text{LSTM}_f(\mathbf{x}_t, \mathbf{h}_{t-1}^{(f)}), \quad \mathbf{h}_t^{(b)} = \text{LSTM}_b(\mathbf{x}_t, \mathbf{h}_{t+1}^{(b)}),$$

yielding a combined hidden state

$$\mathbf{h}_t = [\mathbf{h}_t^{(f)}; \mathbf{h}_t^{(b)}].$$

This allows the model to incorporate future context alongside past context, often beneficial for identifying signals near either terminus of a protein. The final classification follows similarly by applying a dense layer on the concatenated hidden states at the last step or a pooled representation of all time steps.

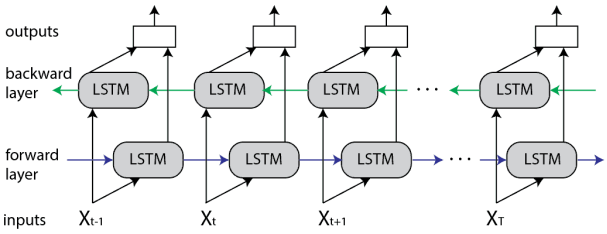


Fig. 6. Bidirectional LSTM Model Architecture.

Attention-based LSTM To concentrate on critical subsequences (e.g., NLS motifs), we add a self-attention mechanism on top of the LSTM hidden states $\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$. Let $\mathbf{h}_t$ be the LSTM's hidden state at position t. We compute a **context vector** $\mathbf{v}$ as:

$$\alpha_t = \frac{\exp(\mathbf{w}_a^\top \tanh(W_a \mathbf{h}_t))}{\sum_{k=1}^{T} \exp(\mathbf{w}_a^\top \tanh(W_a \mathbf{h}_k))}, \qquad \mathbf{v} = \sum_{t=1}^{T} \alpha_t \mathbf{h}_t,$$

where α_t is the attention weight, $\mathbf{w}_a$ and W_a are learnable parameters, and tanh is applied element-wise. The final classification uses $\mathbf{v}$ instead of a simple pooling, allowing the model to highlight residues that are particularly indicative of subcellular localization signals.

LSTM + CNN Hybrid Convolutional networks excel at detecting local sequence motifs, while LSTMs capture global context. Our CNN-LSTM hybrid appends a **2D convolutional** layer(s) before feeding the extracted local features to the LSTM. The CNN is defined by filters $\mathbf{F}_j$ of size $k \times k$:

$$\mathbf{z}_j[t] = \mathrm{ReLU}\left(\sum_{p=0}^{k-1} \sum_{q=0}^{k-1} \mathbf{F}_j \odot \mathbf{X}_{t+p,q} + b_j \right).$$

To further reduce spatial dimensionality and retain dominant features, we apply max pooling:

$$\mathbf{z}_j^{\mathrm{pool}}[t] = \max_{p,q} \mathbf{z}_j[t+p, q].$$

This yields local motif representations $\mathbf{z}_j^{\mathrm{pool}}[t]$, which are then stacked or flattened and fed into an LSTM to capture longer contexts. Such a multi-stage design aims to unify local motif extraction with global sequence modeling. Similar approaches have been used in protein function prediction [9].

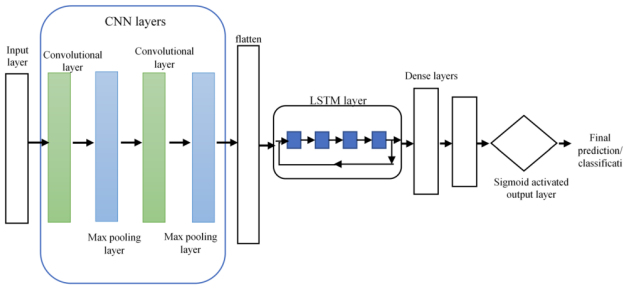


Fig. 7. LSTM-CNN Hybrid Model Architecture.

BiLSTM + CNN + Attention Finally, we combine CNN layers, bidirectional LSTMs, and attention into a comprehensive model (Figure 8). The CNN extracts local patterns, the BiLSTM integrates bidirectional context, and the attention mechanism pinpoints crucial positions in the hidden states. A final dense layer outputs the four class probabilities. This approach seeks to capture the full spectrum of local and long-range signals in protein sequences.

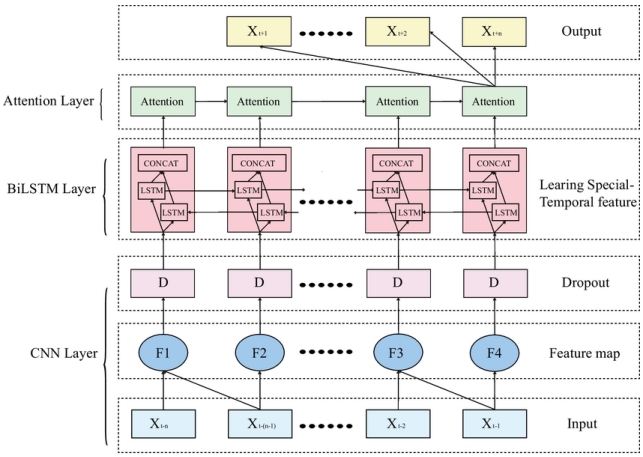


Fig. 8. BiLSTM + CNN + Attention Model Architecture.

Embedding-Based Models

Moving beyond handcrafted features and smaller learned embeddings, we applied pretrained protein language models to generate context-dependent sequence representations. These models differ in token-level outputs, hidden dimensions, and presence or absence of a dedicated [CLS] token. Below, we detail how embeddings were extracted and subsequently fed into multi-layer perceptron (MLP) classifiers.

Embedding Extraction Given a protein sequence $\mathbf{X} = (x_1, x_2, \ldots, x_T)$, each model produces a hidden state $\mathbf{e_t} \in \mathbb{R}^d$ for each residue x_t. The strategy to aggregate these token embeddings differ by model:

- **ProtBERT / ProtAlbert** These models follow a BERT-style architecture, including a [CLS] token at the start of the sequence. The dimension d may vary (e.g., $\mathbf{e_t} \in \mathbb{R}^{4096}$ for ProtAlbert). We typically select the [CLS] vector $\mathbf{e}_{\mathrm{CLS}}$ as the fixed-length embedding $\mathbf{E}$, exploiting the model's design to summarize the entire sequence in that token. Alternatively, mean pooling across all residues is possible but less common with BERT-like models.
- **ProtXLNet** Because XLNet-based architectures do not include a dedicated [CLS] token by default, we mean-pool the residue embeddings:

$$\mathbf{E} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{e_t},$$

where each $\mathbf{e_t} \in \mathbb{R}^{1024}$. This operation yields a single fixed-length $\mathbf{E} \in \mathbb{R}^{1024}$, ensuring consistent input dimensionality to downstream modules.

- **ESM2** (T6 vs. T33) ESM2 models produce token embeddings at various scales: **T6** has a hidden dimension of 320, while **T33** expands to 1280. Both variants include special tokens for positional or classification tasks, but in many use cases, mean pooling over the output tokens is adopted for a stable sequence embedding:

$$\mathbf{E} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{e_t},$$

or, if a special token (e.g., [CLS]) is available, it can be used directly. The smaller T6 model trades some representational capacity for speed and efficiency, whereas T33 typically provides richer embeddings at higher computational cost.

MLP Classifier Once we obtain the aggregated embedding $\mathbf{E} \in \mathbb{R}^d$, we apply a multi-layer perceptron (MLP) to classify proteins into four subcellular locations. Let $\mathbf{h_l}$ denote the hidden layer activations at layer l. The general form is:

$$\mathbf{h_1} = \sigma(\mathbf{W_1 E} + \mathbf{b_1}), \quad \mathbf{h_2} = \sigma(\mathbf{W_2 h_1} + \mathbf{b_2}), \ldots,$$

$$\mathbf{o} = \mathbf{W}_{\text{out}}\mathbf{h_L} + \mathbf{b}_{\text{out}},$$

where σ is a non-linear activation (e.g., ReLU). The **softmax** function is then applied to $\mathbf{o}$ to obtain class probabilities.

- **Shallow MLP (ProtBERT, ProtXLNet, ProtAlbert)** For these models, $\mathbf{E}$ (dimension = hidden embedding size) feeds into:

$$\mathbf{h_1} = \text{ReLU}(\mathbf{W_1 E} + \mathbf{b_1}),$$

followed by dropout and a final linear layer to output four logits. Concretely:

$$\text{Classifier:} \quad \mathbb{R}^d \to \mathbb{R}^{512} \to \mathbb{R}^4.$$

Dropout at 0.3 helps mitigate overfitting, while ReLU provides stable gradient flow.
- **Deeper MLP (ESM2 T6, T33)** For ESM2 embeddings, we employ an additional hidden block and batch normalization:

$$\mathbf{h_1} = \text{BN}(\text{ReLU}(\mathbf{W_1 E}+\mathbf{b_1})), \quad \mathbf{h_2} = \text{BN}(\text{ReLU}(\mathbf{W_2 h_1}+\mathbf{b_2})).$$

Dropout layers with rates of 0.3 and 0.2 are interspersed to prevent overfitting. The final output is:

$$\mathbf{o} = \mathbf{W_3 h_2} + \mathbf{b_3}.$$

Thus, the ESM2 classifier head has three linear layers vs. the two-layer design used for other models. This deeper architecture helps leverage ESM2's richer embeddings, especially in the T33 version (d=1280).

4 Implementation Details

All experiments were conducted on a machine equipped with a GPU (e.g., NVIDIA RTX-series) for accelerating deep neural network training. **Python 3.10.16** was the primary programming language, using a combination of scikit-learn for classical ML models and PyTorch for deep learning architectures.

Data Preprocessing

Prior to model training, we performed several preprocessing steps to reduce model complexity and ensure efficient mini-batching:

- **Sequence Truncation:** As most protein sequences in our dataset are within 2,000 residues, sequences longer than 2,000 were truncated. Specifically, we retained the first 2,400 and the last 100 amino acids, thereby preserving key targeting signals at both the N- and C-termini while removing the middle section.
- **Padding:** For sequences shorter than 2,000 residues, we inserted a special PAD token in the middle to standardize sequence length across the batch in order to maintains the critical terminal regions and facilitates mini-batch training.

Training Pipeline and Cross-Validation

During model training, the training set of 7,458 sequences was split into train and validation subsets (e.g., **80%** train, **20%** validation) and each model was trained on the same training subset and evaluated on the validation set at the end of every epoch. To reduce computational overhead during the initial comparisons, we did not use cross-validation at this stage; rather, we trained each candidate model using this fixed split.

Hyperparameter Tuning

Hyperparameter tuning was conducted using two distinct strategies: **Grid Search for SVM** For SVM, we performed a grid search over predefined parameter sets:

- kernel type $\in$ {RBF}, penalty $C \in \{0.1, 1, 10\}$, gamma $\in \{\text{`'scale''}, \text{`'auto''}, 0.01, 0.1, 1\}$.

The optimal hyper-parameters were selected based on validation accuracy on the 20% validation split.

Bayesian Optimization for DL Models We adopted Bayesian optimization to navigate the larger, or complex hyperparameter spaces. Parameter sets are:

- **Learning rate** (α): searched within $[10^{-5}, 10^{-2}]$.
- **Batch size** ($\in \{32, 64, 128\}$).
- **Number of hidden units** ($\in \{128, 256, 512\}$).
- **Kernel Size** ($\in \{[3, 5, 7], [3, 5, 9], [3, 7, 11]\}$).
- **Embedding size** ($\in \{32, 64, 128\}$).

Bayesian optimization iteratively refined its search by modeling validation performance as a surrogate function over parameter space, aiming to minimize validation loss. We performed manual tuning for transformer-based embeddng models. Details of the final chosen parameters for each model are provided in Appendix D.

Final Training and Testing

By hyperparameter optimization, each candidate model was tuned individually, and the model demonstrating the highest overall performance was chosen for further evaluation. To assess the generalization capability of the selected model, a 5-fold cross validation procedure was conducted to verify

robustness and obtain a more stable estimate of generalization performance. The optimal model was then re-trained on the combined (train + validation) subset of 7,458 sequences. We then used the **held-out test set** of 20 sequences to obtain the final predictions and assess the confidence level based on the class predicted probability.

5 Results & Discussion

Global Model Comparisons

We evaluated the predictive strength and limitations of the following approaches:

- **Classical ML Model**: Support Vector Machine (SVM).
- **Deep Learning Architectures**: LSTM, BiLSTM, LSTM + Attention, LSTM-CNN, and BiLSTM + Attention + CNN.
- **Transformer-based Embeddings**: ProtBERT (+ MLP), ProtAlbert (+ MLP), ProtXLNet (+ MLP), ESM2-T6 (+ MLP), ESM2-T33 (+ MLP).

We analyzed five key metrics: **macro-F1**, **accuracy**, **ROC-AUC**, **precision**, and **recall**. Given the moderate imbalance in our dataset, macro-F1 is particularly useful as it ensures that each class is equally represented in the evaluation, thus providing a fair assessment of performance across both majority and minority classes. Additionally, the confusion matrices for each models can be found in Appendix A which provides supplementary insights on class-specific misclassifications.

Model	Macro-F1	Accuracy	ROC-AUC	Precision	Recall
SVM	0.658	0.659	0.766	0.662	0.667
LSTM	0.726	0.708	0.859	0.719	0.736
BiLSTM	0.738	0.716	0.870	0.746	0.734
LSTM-CNN	0.727	0.707	0.878	0.722	0.734
Attention-LSTM	0.738	0.717	0.864	0.732	0.746
CNN-BiLSTM-Attention	0.733	0.715	0.852	0.730	0.736
ProtBERT + MLP	0.729	0.693	0.889	0.739	0.720
ProtAlbert + MLP	0.778	0.756	0.920	0.789	0.771
ProtXLNet + MLP	0.768	0.749	0.923	0.778	0.762
ESM2-T6 + MLP	0.755	0.722	0.907	0.766	0.747
ESM2-T33 + MLP	**0.834**	**0.819**	**0.938**	**0.846**	**0.834**

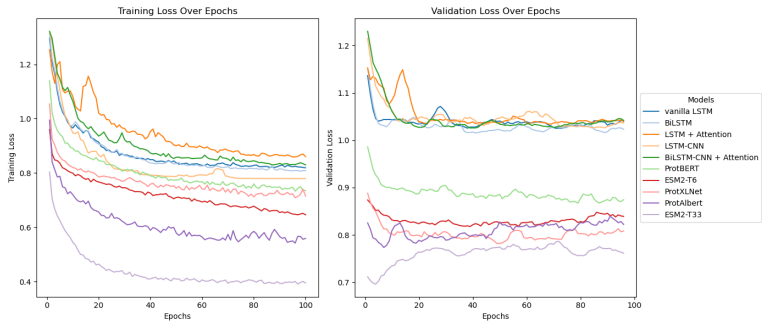
Table 1. Performance metrics of different models (highlighted top model per metric).

The results, summarized in Table 1, reveal that:

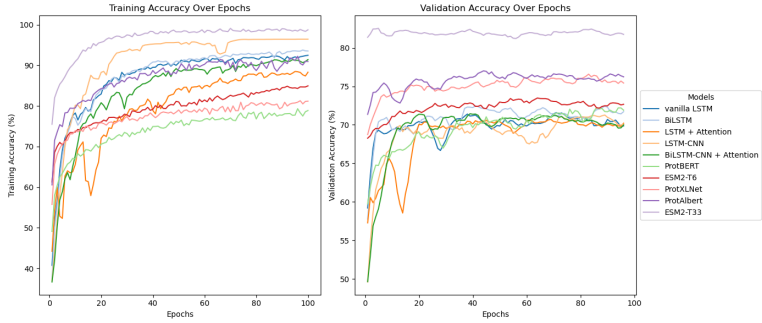
1. **Embedding-based models outperform all others.** Transformer-based embeddings consistently outperform LSTM-based architectures in Macro-F1 (0.75–0.84 vs. 0.71–0.74).
2. **Attention-enhanced LSTMs improve sequence learning.** CNN-BiLSTM-Attention and Attention-LSTM outperformed baseline LSTMs, with macro-F1 scores of 0.733 and 0.738, respectively. The addition of attention mechanisms likely helped focus on critical motifs.
3. **ESM2-T33 emerges as the top performer across most metrics.** With a Macro-F1 of 0.84 and a ROC-AUC of 0.94, ESM2-T33 demonstrates a strong ability to distinguish among the four classes.

Training and Validation Curves

Figure 9 shows training and validation loss and accuracy over 100 epochs:



(a) Training/Validation Loss Curve.



(b) Training/Validation Accuracy Curve.

Fig. 9. Training and Validation Curves Over Epochs Across all Models.

- **Overall Ranking** ESM2-T33 is the clear winner on validation accuracy (~82%) and lowest validation loss (~0.75–0.80). It's followed by ProtAlbert (≈76–77% val-acc), then ProtXLNet (≈75–76%), ESM2-T6 (≈72–73%), and ProtBERT (≈71–72%). All LSTM-based models plateau around 70% with higher loss (~1.03).
- **Generalization Speed** ESM2-T33 and ProtAlbert achieve 90 – 100% training accuracy but still hold a consistent ~10-point gap to validation—indicative of high capacity plus mild overfitting. LSTM models have smaller train—val gaps, except LSTM-CNN, but much lower absolute performance.
- **Stability & Noise** Validation curves for pretrained models (especially ProtAlbert, ESM2-T33) are smoother and steadily improve. In contrast, the LSTM+Attention model shows large oscillations in both val-loss and val-acc (poor stability).

Class-Specific Metrics

To understand where each model excels or misclassify and why, we examined per-class Macro-F1 score across all models:

(a)**Secreted Proteins** (F1 ≈ 0.88 – 0.95)
Among all four localization classes, secreted proteins consistently achieve the highest F1-scores across every model architecture. This near-perfect performance stems from the presence of a **canonical N-terminal signal peptide**—a short, hydrophobic motif whose sequence

features are both highly conserved and spatially restricted to the protein's N-terminus. Even simple recurrent networks readily learn this motif; however, pretrained transformers significantly outperform LSTM variants by leveraging the attention mechanisms that precisely localize and contextualize the signal peptide within each sequence. Consequently, transformers minimize false positives (precision) by distinguishing secretory signals from superficially similar hydrophobic regions, and minimize false negatives (recall) by recognizing variant peptide lengths and flanking residues.

Residual misclassifications primarily involve dual-targeting signals (for example, ER-retention KDEL motifs) or atypical secretion signals, underscoring that remaining errors arise from genuine biological ambiguity rather than model deficiency.

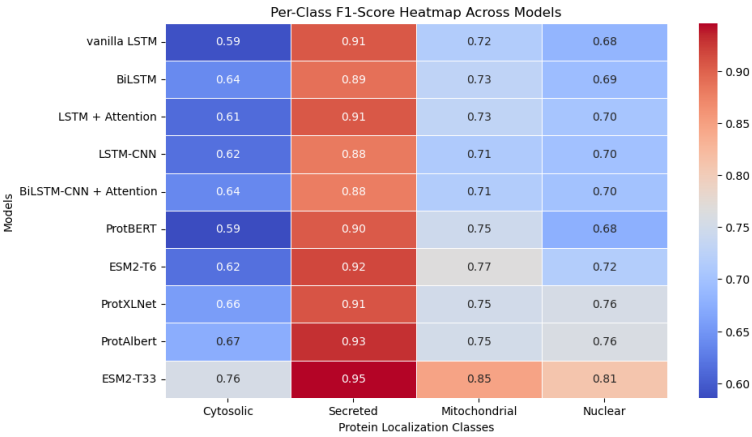


Fig. 10. Per-Class F1-Score Heatmap Across Models.

(b)**Cytosolic Proteins** (F1 ≈ 0.59 – 0.76)

Predicting cytosolic localization presents the greatest challenge because cytosolic assignment is defined by the absence of any targeting motif rather than a positive signature. This negative-evidence problem requires models to learn what a lack of signal looks like in a high-dimensional sequence context.

Transformer architectures outperform LSTMs by encoding global sequence patterns and subtle co-occurrence statistics that differentiate incidental motif-like features (e.g., short basic patches) from true targeting signals. Consequently, large pretrained models achieve superior negative-signal discrimination, reflected in their higher precision and recall.

Persistent confusion with **nuclear localization** arises from incidental nuclear localization signals (NLS) in housekeeping proteins, and with **mitochondrial proteins** when weak amphipathic signals mimic transit peptides.

(c)**Mitochondrial Proteins** (F1 ≈ 0.71 – 0.85)

Mitochondrial localization yields intermediate performance because transit peptides are structurally conserved as amphipathic helices yet display greater sequence variability and less positional constraint than secretory signals. High-capacity transformers capture the physicochemical signature of amphipathicity and integrate long-range residue interactions, boosting both precision and recall. In contrast, LSTM-based models plateau at lower performance, lacking the ability to jointly model sequence context and secondary structure propensity.

Misclassification arises chiefly when transit peptides deviate from canonical patterns—either in length or charge distribution—or overlap in hydrophobic content with secretory signals, particularly for mid-sized pretrained models such as ProtAlbert and ProtXLNet.

(d)**Nuclear Proteins** (F1 ≈ 0.68 – 0.81)

Nuclear localization produces moderate F1-score because nuclear localization signals are short, basic patches that frequently occur by chance in cytosolic proteins. Effective discrimination therefore demands modeling the broader sequence context around putative NLS motifs to distinguish functional targeting signals from incidental basic residues. Transformers excel at this task, leveraging attention layers to learn residue co-occurrence patterns and higher-order sequence dependencies that indicate bona fide nuclear targeting. Smaller models like LSTMs under-perform due to limited contextual capacity, resulting in persistent false positives and false negatives.

In summary, classes with unique signal peptides (**secreted, mitochondrial**) are easier to classify, while **cytosolic/nuclear** distinction remains the toughest subtask. Deeper embeddings seem to mitigate confusion by capturing subtle sequence or structural signals that differentiate the two compartments. Per-Class Precision & Recall heatmap across models can be found in Appendix B for further details.

Selecting the Best Model

As we defined macro-F1 as the primary criterion for selecting the best-performing model, ensuring that less frequent compartments are equally weighted. We obsered that ESM2-T33 achieves the highest Macro-F1 of 0.834, along with a strong ROC-AUC (0.938) and the highest accuracy (0.819). ProtAlbert and ProtXLNet follow as the next best embedding approaches, but with Macro-F1 scores of 0.78 and 0.77 respectively. Thus, **ESM2-T33** stands out as our best model.

Cross-Validation Results of the Best Model

We further validated ESM2-T33 via 5-fold cross-validation (Table 2). The overall mean validation accuracy is 0.815, and macro-F1 is 0.833, indicating robust performance across folds. While some variability appears (Fold 5 dips slightly), ROC-AUC remains consistently high (0.932–0.943), reinforcing ESM2-T33's reliability.

Fold	Mean Val Acc	F1-Score	Precision	Recall	ROC-AUC
Fold 1	0.819	0.842	0.841	0.843	0.942
Fold 2	0.814	0.827	0.835	0.822	0.938
Fold 3	0.819	0.836	0.832	0.839	0.943
Fold 4	0.824	0.844	0.846	0.843	0.941
Fold 5	0.799	0.816	0.823	0.810	0.932
Overall	**0.815**	**0.833**	**0.835**	**0.832**	**0.939**

Table 2. ESM2-T33 Performance Across 5-Fold Cross-Validation

The mean confusion matrix (Figure11) shows that cytosolic vs. nuclear overlap remains the largest challenge, with a notable fraction of cytosolic proteins labeled as nuclear (and vice versa). In contrast, secreted proteins exhibit minimal confusion, reflecting the model's ability to detect distinct signal peptides. Mitochondrial proteins occasionally appear as cytosolic, likely due to the short, amphipathic nature of transit peptides. Despite these errors, each class attains a substantial number of correct predictions, underscoring ESM2-T33's strong overall discrimination. This aligns with previous findings that secreted and mitochondrial classes are easier to identify.

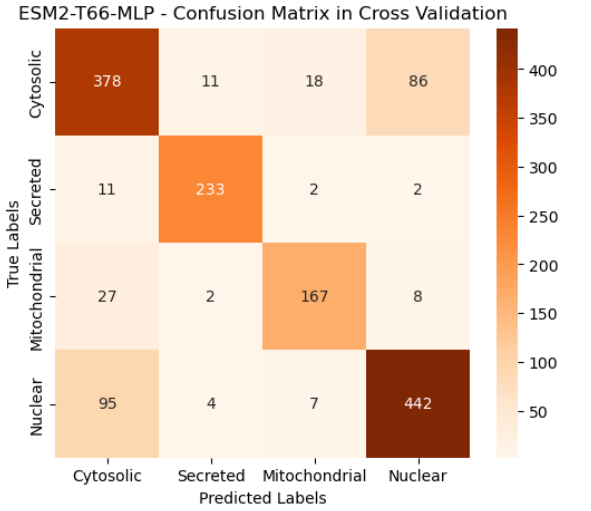


Fig. 11. ESM2-T66-MLP Confusion Matrix during Cross Validation.

Final Test Set Performance

Upon finalizing ESM2-T33 model, we retrained on all 7,458 training sequences and evaluated on the 20-sequence blind set. Table 3 shows each protein's predicted class and a confidence score derived from the model's softmax probabilities. The model predicts predominantly cytoplasmic proteins (8 instances), possibly reflecting a cytosol-bias if genuine nuclear proteins exhibit partial motifs or ambiguous signals.

A majority of predictions are labeled "Confidence High", with only three sequences ($SEQ02$, $SEQ10$ & $SEQ14$) receive "Confidence Medium". Overconfidence is a known issue in deep neural networks, especially large-scale transformers, where learned motifs can inflate probability scores. Future work may mitigate such risks via calibration methods (e.g., temperature scaling [23]) or Bayesian neural network and Monte Carlo dropout that provide more nuanced uncertainty estimates and confidence intervals.

Feature Importance Analysis

We employed Integrated Gradients (IG)[24] to identify which embedding dimensions of the ESM2-T33 model most strongly influence the final predictions. IG was computed for each sample using the predicted class logit, providing a gradient-based attribution of input feature relevance.

Sequence ID	Predicted Class	Confidence
SEQ01	Cytoplasm	Confidence High
SEQ02	Nucleus	Confidence Medium
SEQ03	Mitochondrion	Confidence High
SEQ04	Cytoplasm	Confidence High
SEQ05	Cytoplasm	Confidence High
SEQ06	Mitochondrion	Confidence High
SEQ07	Secreted	Confidence High
SEQ08	Mitochondrion	Confidence High
SEQ09	Secreted	Confidence High
SEQ10	Secreted	Confidence Medium
SEQ11	Cytoplasm	Confidence High
SEQ12	Cytoplasm	Confidence High
SEQ13	Secreted	Confidence High
SEQ14	Nucleus	Confidence Medium
SEQ15	Cytoplasm	Confidence High
SEQ16	Nucleus	Confidence High
SEQ17	Cytoplasm	Confidence High
SEQ18	Nucleus	Confidence High
SEQ19	Cytoplasm	Confidence High
SEQ20	Secreted	Confidence High

Table 3. Predicted Class and Confidence for Protein Sequences

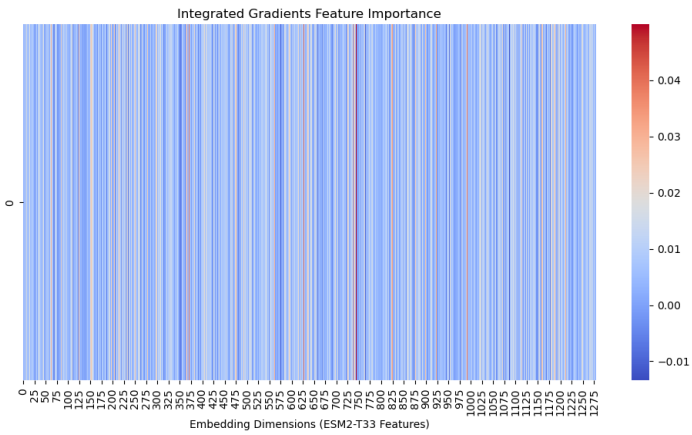


Fig. 12. Integrated Gradients Feature Importance (Averaged).

Global Integrated Gradients Figure 12 shows a heatmap of the averaged IG attributions across all 1280 embedding dimensions. A large majority of dimensions exhibit near-zero attributions, indicating minimal impact on the final classification.

However, several standout dimensions have substantially higher IG values (positive or negative), suggesting they capture key sequence signals. Table 4 lists the top 10 most influential dimensions based on mean absolute IG scores. Because we lack detailed protein-level metadata, the biological meaning of these dimensions remains unclear, and further experimentation would be required to ascertain whether they map to specific biochemical features or structural motifs.

Rank	Dimension (Score)
1	Dim 736 (0.1401)
2	Dim 829 (0.1309)
3	Dim 832 (0.1234)
4	Dim 825 (0.1200)
5	Dim 327 (0.1196)
6	Dim 467 (0.1179)
7	Dim 13 (0.1135)
8	Dim 528 (0.1102)
9	Dim 1249 (0.1095)
10	Dim 362 (0.1079)

Table 4. Top 10 Dimensions with IG Scores.

Class-Wise IG Comparsions To see whether different classes rely on distinct dimensions, we computed IG separately for each subcellular localization. As shown in Figure 13, most classes share a common set of influential dimensions with only minor class-specific peaks.

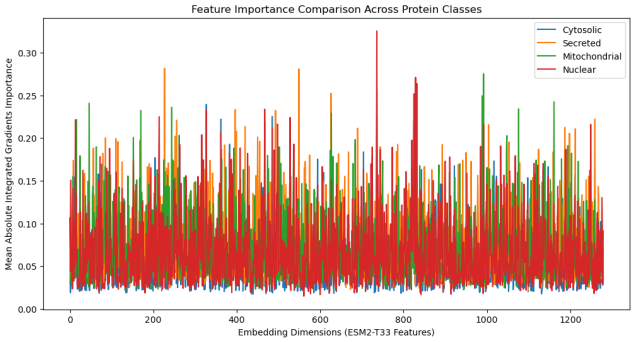


Fig. 13. IG Feature Imporatnce Comparsion Across 4 Protein Classes.

Indeed, a cosine similarity heatmap (Figure14) reveals a strong overlap (0.93) between cytosolic and nuclear proteins, reflecting the known difficulty in distinguishing these two classes by the absence or subtlety of targeting signals. In contrast, secreted and mitochondrial classes have lower similarity (0.76), aligning with the fact that both harbor clearer N-terminal motifs—signal peptides or transit peptides, respectively.

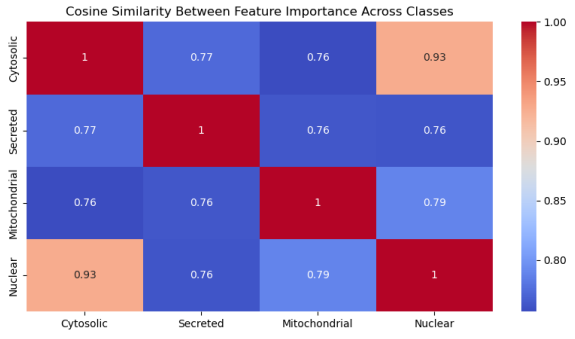


Fig. 14. Cosine Similarity Heatmap Between IG Importance Score Across Classes.

t-SNE on All Embedding Dimensions

Lastly, a t-SNE projection[25] of the full 1280 dimensional embeddings (Figure 15) shows partial clustering: secreted and mitochondrial proteins occupy relatively distinct regions, while nuclear and cytosolicoverlap significantly. This finding corroborates earlier metrics showing that nuclear vs. cytoplasmic classification is the main source of confusion. Notably, when we restricted the model to just the top 10 IG-ranked dimensions (see Appendix E), overall class separation deteriorated, illustrating the importance of broad embedding features for robust PSL prediction.

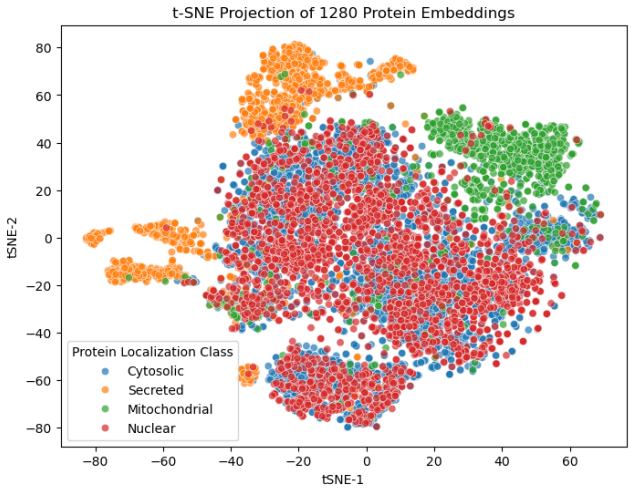


Fig. 15. t-SNE Project of 1280 ESM2-T33 Embedding Dimensions.

In summary, **IG-based attributions** reveal that only a subset of ESM2-T33's embedding dimensions strongly influence classification, but these dimensions alone are insufficient to separate all classes in low-dimensional space. Most classes appear to rely on a shared core of features, while specialized motifs for secreted or mitochondrial proteins diverge more starkly—explaining their higher classification performance compared to cytosol/nucleus overlap.

6 Conclusion

In this study, we systematically compared classical ML (SVM), LSTM-based deep learning (BiLSTM, LSTM+Attention, CNN hybrids), and transformer-based embeddings (ProtBERT, ProtXLNet, ProtAlbert, ESM2) for protein subcellular localization across four compartments: cytoplasmic, secreted, mitochondrial, and nuclear. ESM2-T33 consistently outperformed all other models on macro-F1, accuracy, and cross-validation metrics, underscoring the benefits of large-scale, pretrained embeddings for capturing subtle sequence features.

A breakdown of class-specific results showed that secreted proteins are the easiest to predict (due to distinct signal peptides), whereas cytosolic vs. nuclear classification remains the toughest. This challenge stems from the lack of a definitive negative signal for cytoplasmic proteins and the short, easily confused NLS motifs in nuclear proteins. While integrated gradients provided some interpretability, deeper

insights, especially for ambiguous residues, remain elusive. Furthermore, overfitting risks in large transformer models necessitate additional regularization or data augmentation.

Moving forward, strategies such as confidence calibration (e.g., Platt scaling, Bayesian inference) and multi-label architectures could mitigate overconfidence and capture real-world scenarios where proteins occupy multiple compartments. Overall, our findings emphasize the superior efficacy of large, pretrained embeddings (notably ESM2-T33) but also highlight the ongoing need to address interpretability and uncertainty for robust applications in both research and clinical settings.

References

1. B. Franz Lang, Natacha Beck, Samuel Prince, Matt Sarrasin, Pierre Rioux, and Gertraud Burger. Mitochondrial genome annotation with mfannot: a critical analysis of gene identification and gene model prediction. *Frontiers in Plant Science*, 14:1222186, 2023.
2. M. Anteghini, A. Haja, V.A.P. Martins Dos Santos, L. Schomaker, and E. Saccenti. Organelx web server for sub-peroxisomal and sub-mitochondrial protein localization and peroxisomal target signal detection. *Computational and Structural Biotechnology Journal*, 21:128–133, 2022.
3. H. Chen, Y. Cai, C. Ji, G. Selvaraj, D. Wei, and H. Wu. Adappi: Identification of novel protein functional modules via adaptive graph convolution networks in a protein-protein interaction network. *Briefings in Bioinformatics*, 24(1):bbac523, 2023.
4. Z. Liao, G. Pan, C. Sun, and J. Tang. Predicting subcellular location of protein with evolution information and sequence-based deep learning. *BMC Bioinformatics*, 22(Suppl 10):515, October 22 2021.
5. Z. Hou, Y. Yang, H. Li, K. C. Wong, and X. Li. ideepsubmito: Identification of protein submitochondrial localization with deep learning. *Briefings in Bioinformatics*, 22(6):bbab288, November 2021.
6. Kaleel M, Zheng Y, Chen J, Feng X, Simpson JC, Pollastri G, and Mooney C. Sclpred-ems: subcellular localization prediction of endomembrane system and secretory pathway proteins by deep n-to-1 convolutional neural networks. *Bioinformatics*, 36(11):3343–3349, June 2020.
7. H. Cong, H. Liu, Y. Cao, Y. Chen, and C. Liang. Multiple protein subcellular locations prediction based on deep convolutional neural networks with self-attention mechanism. *Interdisciplinary Sciences: Computational Life Sciences*, 14(2):421–438, June 2022. Epub 2022 Jan 23.
8. M. Anteghini, V. Martins Dos Santos, and E. Saccenti. In-pero: Exploiting deep learning embeddings of protein sequences to predict the localisation of peroxisomal proteins. *International Journal of Molecular Sciences*, 22(12):6409, 2021.
9. Søren Kaae Sønderby, Casper Kaae Sønderby, Henrik Nielsen, and Ole Winther. *Convolutional LSTM Networks for Subcellular Localization of Proteins*, page 68–80. Springer International Publishing, 2015.
10. Nadav Brandes, Dan Ofer, Nadav Rappoport, and Michal Linial. ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, April 2022.
11. Zeyu Wang, Tao Lin, Xiaoli Yang, Yanchun Liang, and Xiaohu Shi. Protein subcellular localization prediction by combining protbert and bigru. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 86–89, 2022.
12. Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*, 2019.
13. Tianqi Zhang, Jing Gu, Zhe Wang, Chuan Wu, Yadong Liang, and Xiaowei Shi. Protein subcellular localization prediction model based on graph convolutional network. *Interdisciplinary Sciences: Computational Life Sciences*, 14(4):937–946, December 2022.
14. K. Jha, S. Karmakar, and S. Saha. Graph-bert and language model-based framework for protein-protein interaction identification. *Scientific Reports*, 13(1):5663, 2023.
15. José J. Almagro Armenteros, Casper K. Sønderby, Søren K. Sønderby, Henrik Nielsen, and Ole Winther. Deeploc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, 33(21):3387–3395, Nov 1 2017. Erratum in: Bioinformatics. 2017 Dec 15;33(24):4049. doi: 10.1093/bioinformatics/btx548.
16. S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721–728, August 2001.
17. A. Garg, M. Bhasin, and G. P. Raghava. Support vector machine-based method for subcellular localization of human proteins using amino acid compositions, their order, and similarity search. *Journal of Biological Chemistry*, 280(15):14427–14432, April 2005. Epub 2005 Jan 12.
18. A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, October 2022. Epub 2022 Sep 14.
19. Armin Behjati, Fatemeh Zare-Mirakabad, Seyed Shahriar Arab, and Abbas Nowzari-Dalini. Protein sequence profile prediction using protalbert transformer. *bioRxiv*, 2021.
20. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.
21. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

22. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
23. Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.
24. Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328, 2017.
25. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

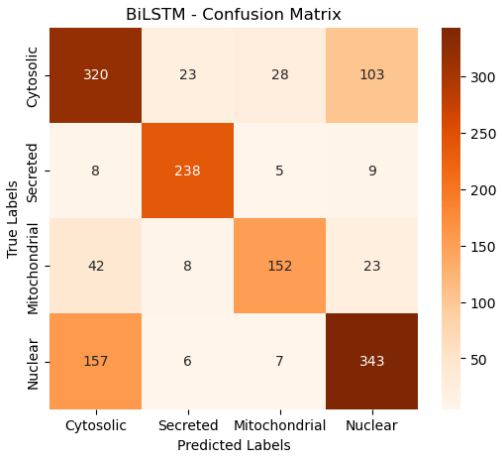


Fig. 18. BiLSTM Confusion Matrix (Counts) in Validation

A Appendix A: Confusion Matrices for All Models

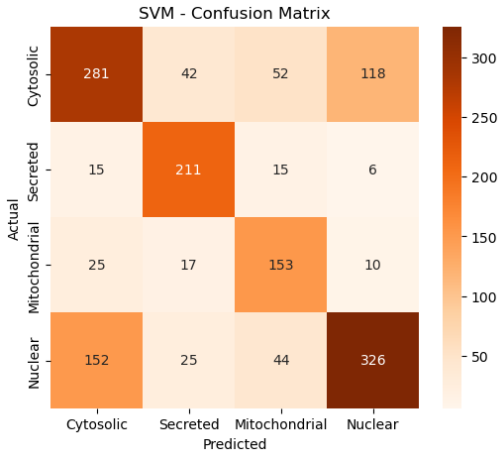


Fig. 16. SVM Confusion Matrix (Counts) in Validation

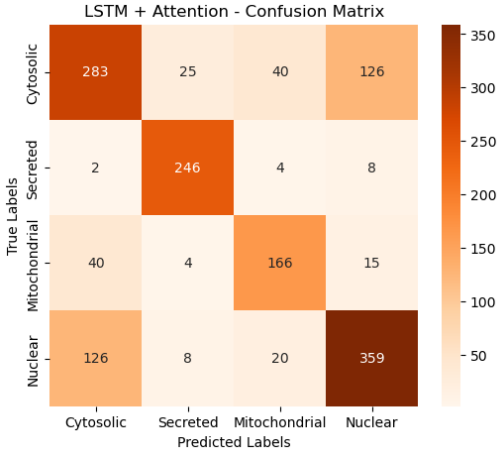


Fig. 19. LSTM+Attention Confusion Matrix (Counts) in Validation

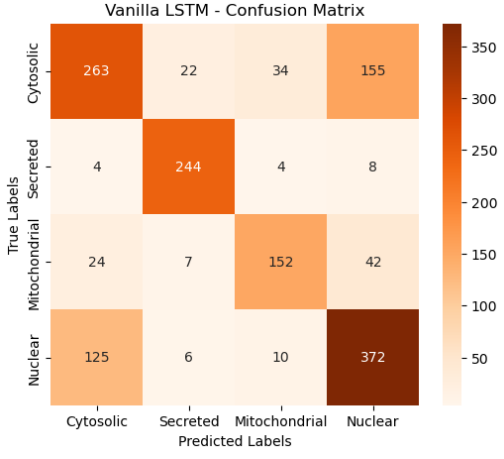


Fig. 17. Vanilla LSTM Confusion Matrix (Counts) in Validation

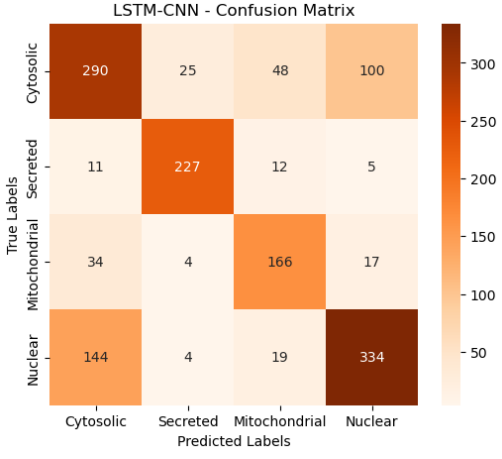


Fig. 20. LSTM-CNN Confusion Matrix (Counts) in Validation

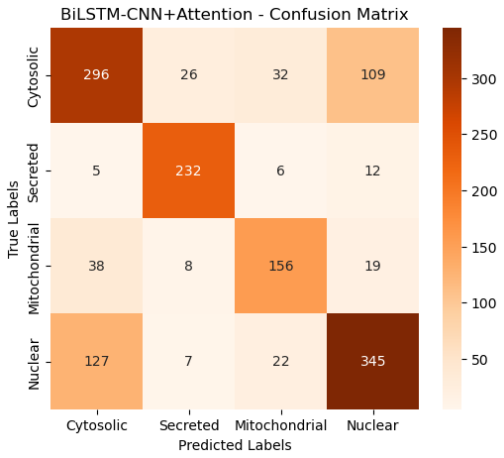


Fig. 21. BiLSTM-CNN+Attention Confusion Matrix (Counts) in Validation

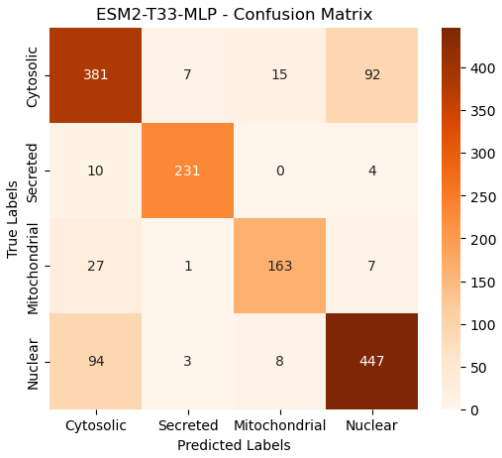


Fig. 24. ESM2-T33-MLP Confusion Matrix (Counts) in Validation

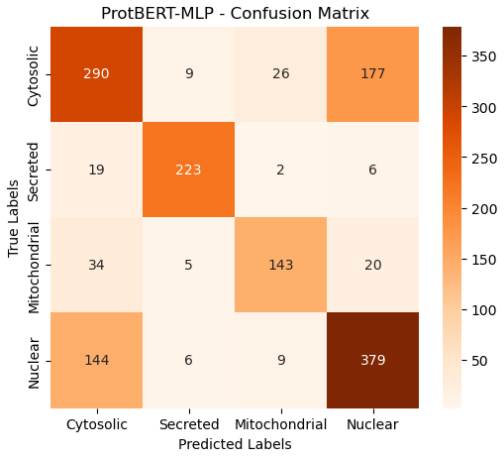


Fig. 22. ProtBERT-MLP Confusion Matrix (Counts) in Validation

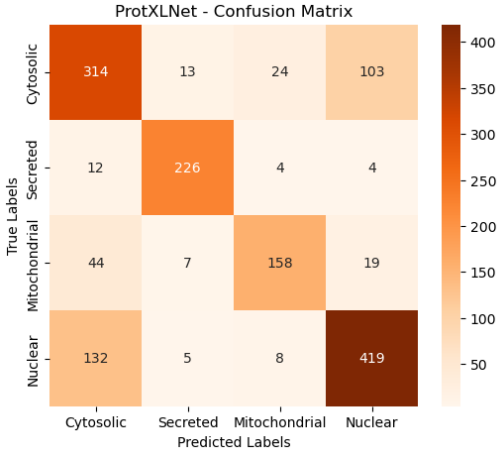


Fig. 25. ProtXLNet-MLP Confusion Matrix (Counts) in Validation

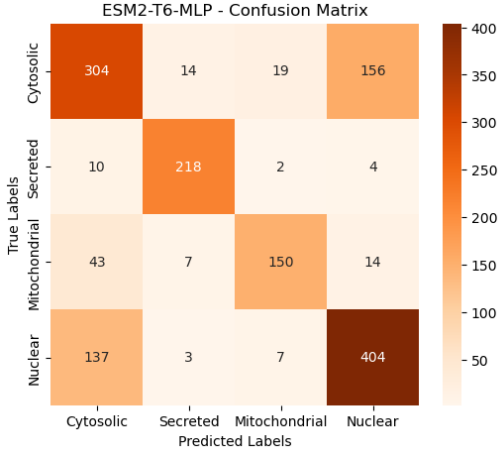


Fig. 23. ESM2-T6-MLP Confusion Matrix (Counts) in Validation

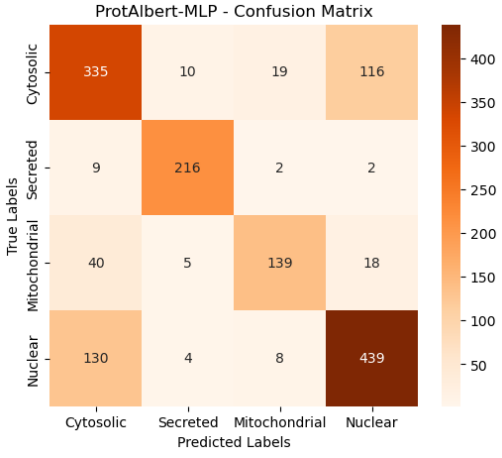


Fig. 26. ProtAlbert-MLP Confusion Matrix (Counts) in Validation

B Appendix B: Per-class Precision & Recall Heatmaps Across Models

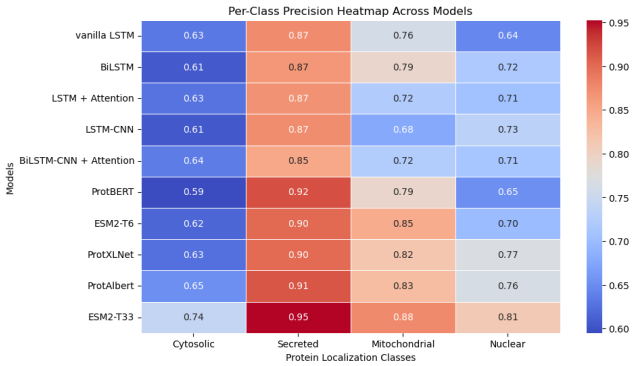


Fig. 27. Per-Class Precision Heatmap Across Models.

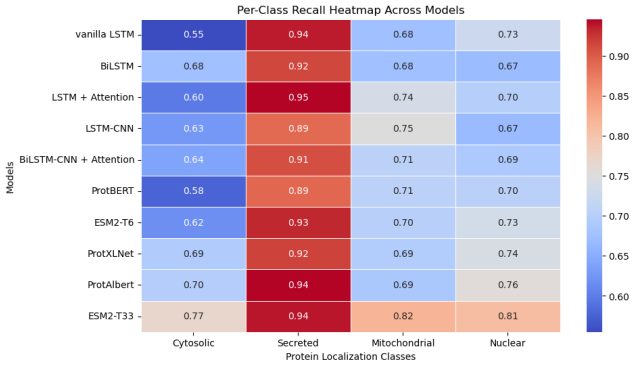


Fig. 28. Per-Class Recall Heatmap Across Models.

C Appendix C: Code Technical Summary

The source code can be found in the following GitHub repository: https://github.com/Melodiepo/protein-subcellular-localization-prediction.

C.1 Data Loading

- `data_fasta.py`: Parses raw FASTA files into dictionaries of sequences, lengths, and labels.
- `pickle_data.py`: Converts these dictionaries into pickled training/test sets, with a unified `get_data` interface.
- `config.py`: Centralizes constants (file paths, max lengths, class names).
- `preprocessing.py`: Builds token dictionaries (`build_dictionary`), encodes sequences into padded arrays, and provides `summary_stats` for exploratory counts.

C.2 Feature Extraction

- `feature_extraction.py`: Cleans sequences, computes amino acid compositions, detects motifs, and assembles all features (length, composition, molecular properties) into 71-dimensional vectors.

C.3 Exploratory Data Analysis (EDA)

- Histograms for sequence-length distributions.
- `summary_stats`: Frequency tables and descriptive stats.
- Correlation heatmaps and box/bar plots to visualize distributions (e.g., isoelectric points, motif counts).

C.4 ML/DL Model Fitting

- `experiments.py`: Houses `experiment_`*functions* (e.g., LSTM, BiLSTM, CNN-LSTM) with a shared `_run_experiment` method for shuffling, k-fold splits, and logging.
- `trainer.py` (`run` class): Manages batching, training/validation loops, confusion matrices, and optional attention tracking.
- `model_definitions.py`: Defines architectures (`attentionLSTM`, `CNN_lstm`, etc.) and optional attention modules.

C.5 Transformer-Based Embeddings

- `ProteinDataset`: Tokenizes sequences for transformer models (ProtBERT, ESM2, etc.).
- `compute_and_save_embeddings`: Runs models once, storing hidden-state vectors as `.pt` files.
- `PrecomputedEmbeddingDataset`: Wraps saved embeddings in a standard `Dataset` for MLP training.
- `train_classifier`: Trains a simple MLP on loaded embeddings, logging metrics (loss, accuracy, confusion matrix).

C.6 Model Evaluation

- Aggregates pickled results via a model-to-file mapping.
- `extract_metrics`: Normalizes result tuples (4-/8-element) for plotting curves.
- Per-class metrics (precision, recall, F1) stored in DataFrames for heatmaps.
- Overall metrics (macro-F1, ROC-AUC) compiled into summary tables.

C.7 Cross-Validation, Retraining, and Blind Test

- 5-fold via `experiment_esm_mlp_kfold`, saves average confusion matrix.
- Test embeddings generated similarly, then predicted with `train_on_full_data_and_test_esm`, returning class probabilities and confidence scores.

C.8 Feature Analysis

- Integrated Gradients (`Captum`): Identifies dimension-wise attributions for ESM2 embeddings.
- Saliency and permutation importance highlight critical features differently (gradient magnitude vs. shuffled accuracy drop).
- t-SNE: 2D visualization of either full embeddings or top IG dimensions.

D Appendix D: Hyperparameter Results

Table 5. Optimal Hyperparameters from SVM via Grid Search

Hyperparameter	Optimal Value
Kernel	rbf
C	1
Gamma	scale
Class Weight	balanced
Decision Function Shape	ovr
Probability	True
Random State	42
Shrinking	True
Tolerance (tol)	0.001

Table 6. Optimal Hyperparameters of LSTM Models Selected via Bayesian Optimization

Model	Hidden Size	Embedding Size	Learning Rate	Kernel Size	Bidirectional	Attention
Vanilla LSTM	256	128	4.24×10^{-4}	N/A	No	No
BiLSTM	256	128	2.97×10^{-4}	N/A	Yes	No
LSTM + Attention	512	64	3.18×10^{-4}	N/A	Yes	Yes
CNN-LSTM	512	128	4.20×10^{-4}	[3, 5, 7]	No	No
CNN-BiLSTM + Attention	256	128	1.85×10^{-4}	[3, 5, 9]	Yes	Yes

Table 7. Optimal Hyperparameters for Embedding-Based Models via Manual Tuning

Model	Embedding Size	Learning Rate	Protein Language Model Parameters
ProtBERT	1024	0.002	12 layers, 768 hidden, 12 attention heads, 110M params
ESM2-T6	320	0.002	6 layers, 320 hidden, 20M params
ESM2-T33	1280	0.002	33 layers, 1280 hidden, 650M params
ProtXLNet	1024	0.002	24 layers, 1024 hidden, 16 attention heads, 409M params
ProtALBERT	4096	0.002	12 layers, 4096 hidden, 64 attention heads, 223M params

E Appendix E: t-SNE Visualization with Reduced Embeddings

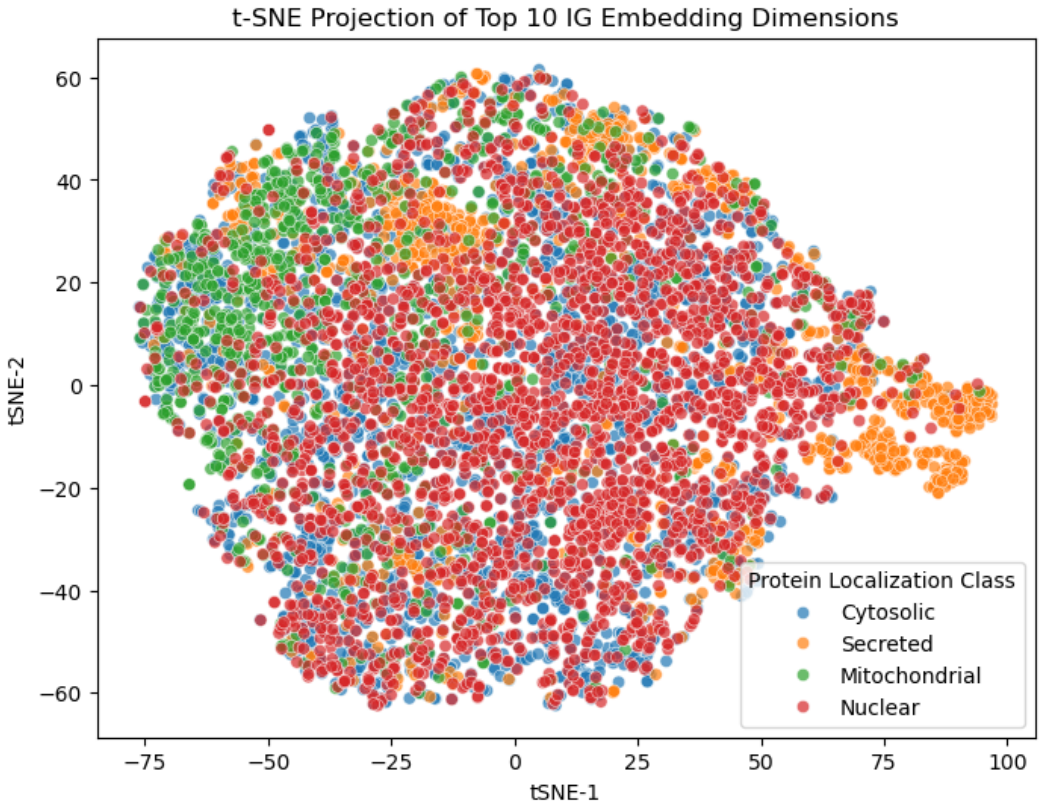


Fig. 29. t-SNE Project of Top 10 IG Embedding Dimensions.