Briefings in Bioinformatics, 2025, pp. 1–14

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

Genomic Language Models (gLMs) decode bacterial genomes for improved Gene Prediction and Translation Initiation Site identification

Genereux Akotenou[1] and Achraf El Allali[1,*]

[1]Bioinformatics Laboratory, College of Computing, University Mohammed VI Polytechnic, Lot 660, Hay Moulay Rachid, Ben Guerir , 43150, Morocco
[*]Corresponding author. achraf.elallali@um6p.ma
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Accurate bacterial gene prediction is essential for understanding microbial functions and advancing biotechnology. Traditional methods based on sequence homology and statistical models often struggle with complex genetic variations and novel sequences due to their limited ability to interpret the "language of genes". To overcome these challenges, we explore Genomic Language Models (gLMs) —inspired by Large Language Models in Natural Language Processing— to enhance bacterial gene prediction. These models learn patterns and contextual dependencies within genetic sequences, similar to how LLMs process human language. We employ transformers, specifically DNABERT, for bacterial gene prediction using a two-stage framework: first, identifying Coding Sequence (CDS) regions, and then refining predictions by identifying the correct Translation Initiation Sites (TIS). DNABERT is fine-tuned on a curated set of NCBI complete bacterial genomes using a k-mer tokenizer for sequence processing. Our results show that GeneLM significantly improves gene prediction accuracy. Compared to the leading prokaryotic gene finders, Prodigal, GeneMark-HMM and Glimmer, and other recent deep learning methods, GeneLM reduces missed CDS predictions while increasing matched annotations. More notably, our TIS predictions surpass traditional methods when tested against experimentally verified sites. GeneLM demonstrates the power of gLMs in decoding genetic information, achieving state-of-the-art performance in bacterial genome analysis. This advancement highlights the potential of language models to revolutionize genome annotation, outperforming conventional tools and enabling more precise genetic insights. The GeneLM tool and source code are publicly available at: github.com/Bioinformatics-UM6P/GeneLM

Key words: Gene prediction, Translation Initiation Site, Large Language Models, Genomic Language Models, BERT, Transformers

INTRODUCTION

Bacterial gene prediction has been a long-standing focus in computational genomics, with numerous tools developed to tackle this challenge. Although significant progress has been made, gene annotation remains an evolving problem, particularly in the accurate identification of coding sequences and translation initiation sites. Traditional gene prediction methods such as Prodigal [14], Glimmer [9], and GeneMark [5] have been widely used for bacterial genome annotation. These tools rely on statistical models, heuristic-based rules, and sequence homology to infer gene structures, but limitations persist. One of the key issues in genome annotation is the variability in composition and organization. High-GC genomes, for example, pose unique challenges due to an increased number of potential open reading frames (ORFs) and ambiguous start codon selection, leading to a decline in prediction accuracy. Furthermore, annotation of Translation Initiation Site remains a difficult task, as start codon selection is influenced by diverse regulatory mechanisms that vary across species. Although existing tools such as TiCO [23] and TriTISA [13] have been developed to refine TIS predictions, these approaches miss several TIS predictions when tested using experimentally verified datasets. Another major challenge lies in the balance between sensitivity and specificity in gene prediction. Traditional methods often overpredict, frequently flagging numerous short ORFs that lack experimental validation. Moreover, genome sequencing analyses have revealed that many essential genes identified through transposon sequencing approaches contain false positives due to genome deletions [18], highlighting the need for methods that improve precision without sacrificing real gene annotations.

Recent advances in Artificial Intelligence and Machine Learning have revolutionized sequence analysis by enabling models to extract meaningful features from vast genomic datasets. Several deep learning-based models have been

proposed for gene prediction, including convolutional neural networks (CNNs) [2, 3]. Self-supervised learning, a technique successfully applied in Natural Language Processing, has demonstrated remarkable capability in understanding sequential patterns and dependencies. Inspired by this progress, Genomic Language Models (gLMs) have emerged as a promising solution, treating DNA sequences as structured linguistic data to enhance gene annotation [4]. Using deep learning architectures, these models can go beyond traditional sequence alignment, capturing both local and global relationships within DNA sequences. Among these architectures, transformers, particularly Bidirectional Encoder Representations from Transformers (BERT) [11] have shown significant promise in sequence-based tasks. Unlike conventional gene prediction models, which rely on predefined feature sets, BERT-based gLMs dynamically learn sequence representations through self-attention mechanisms [25]. This capability allows them to infer gene structures with greater accuracy by modeling long-range dependencies within bacterial genomes. The application of such models offers a path toward more adaptive and precise genome annotation.

To address these challenges, we introduce a transformer-based genomic language model suited for bacterial gene prediction. We use a BERT-based architecture to tackle sequence classification tasks in Prokaryotes gene prediction. To address these tasks, we go through a two-stage classification process. As illustrated in Supplementary Figure S1, given a genome, we have many ORFs that can be extracted. Among those ORFs, some of them cover non-coding regions, while others correspond to coding sequences. In the first step, we classify the coding sequences. Once this is done, in the second step, we focus on the TIS within the coding region. Specifically,

- A BERT-based genomic language model is developed to identify Coding Sequence regions and classify Translation Initiation Sites in Prokaryotes genomes.
- The model is initially trained using self-supervised learning in human genomic datasets to learn DNA fragment representations. It is then adapted for gene annotation through a two-stage pipeline: first, a model classifies ORFs into CDS and non-CDS regions, followed by a refinement stage where a second model identifies true TIS sites.
- Comparative evaluations are conducted against traditional gene prediction tools, assessing improvements in precision, recall, and scalability. A case study on verified bacterial genomes is performed to demonstrate the applicability and robustness of our framework in real-world genomic annotation tasks.
- An analysis is performed to interpret the decision-making process of the model, providing insight into its reasoning, an essential aspect of genomic research.

MATERIALS AND METHODS

Data Collection

To develop a robust and generalized model, it is essential to curate a comprehensive and high-quality dataset. In this study, bacterial genomic data were obtained from the NCBI archive via the GenBank database. Specifically, the assembly summary file was downloaded from the FTP site at ftp.ncbi.nih.gov/genomes/genbank/bacteria. This file, approximately 1GB in size, provides extensive information on bacterial genomes, including annotations, assembly details, and metadata. To ensure dataset quality, only genomes with complete assembly status were considered and filtering was applied to retain only those classified under the "reference genome" category. This resulted in a dataset comprising `5,745 complete and annotated genomes`, representing `4,823 unique organisms`. For each genome in the filtered dataset, two essential files were retrieved: the `genome.fna` file, which contains the complete nucleotide sequences in FASTA format, and the `annotation.gff` file, which provides detailed gene annotations, including coding sequence information. The GFF file delineates genomic features, offering structured annotation data crucial for downstream processing and model training.

Data Processing

To construct high-quality datasets for the translation initiation site and coding sequence classification tasks, we developed a multi-stage processing pipeline. This pipeline extracts open reading frames from bacterial genomes, assigns labels based on genomic annotations and balances the dataset to ensure robust model training.

To extract ORFs, we scanned the forward and reverse strands of each genome sequence using **ORFipy** [22], a fast and flexible Python-based tool for ORF extraction, to identify potential ORFs based on specific start and stop codons. We filter ORFs that began with one of the start codons (`ATG`, `TTG`, `GTG`, or `CTG`) and terminated at a stop codon (`TAA`, `TAG`, or `TGA`). Nested overlapping ORFs were retained to provide comprehensive genomic coverage. Each extracted ORF was subsequently assigned a label by comparing its genomic coordinates with the CDS annotations in the corresponding GFF file. We then labeled two types of datasets, one for CDS and the other for TIS classification. The CDS dataset comprises CSV files containing nucleotide sequences with a maximum length of 510, each labeled positive (1) or negative. A positive label indicates that the sequence represents the longest truncated ORFs whose start or end positions align with an annotated CDS in the GFF reference file. On the other hand, the TIS dataset includes only sequences from ORFs that match a CDS sequence in the reference file and sequences are assigned a binary label, where 1 represents a true translation initiation site (TIS). To capture sequence context, each TIS-centered sequence includes 30 nucleotides upstream and downstream, resulting in a total length of 60 nucleotides. The overall annotation process of both datasets is illustrated in Supplementary Figure S2.

To ensure class balance and mitigate potential biases during model training, different sampling strategies were applied to the CDS and TIS datasets. For the CDS dataset, negative samples were downsampled based on sequence length to match the distribution of the positive class. This approach increases the difficulty of classification, encouraging the model to learn discriminative features beyond sequence length for distinguishing CDS from non-CDS regions. In contrast, for the TIS dataset, where all sequences have the same fixed length, random undersampling was performed to achieve class balance without introducing additional biases. The dataset was partitioned into training, testing, and evaluation sets. For the CDS dataset, the class-balanced splits consist of `14,975,672` sequences for training, `2,181,188` for testing, and `4,253,562` for evaluation. Similarly, for the TIS dataset, the partitioning resulted in `14,544,028` sequences for training, `2,199,192` for testing, and `4,136,340` for evaluation.

Tokenization & Embeddings

To process DNA sequences effectively during training, we employ the `k-mer encoding` technique used in the DNABERT [15] framework. In this method, the DNA sequence is split into overlapping substrings of length k, known as *k-mers*. Each k-mer serves as a discrete token, like words in natural language processing. According to the DNABERT paper, state-of-the-art performance was achieved using $k = 6$. Therefore, for our experimentation, we use a $k = 6$ encoding scheme. Given a DNA sequence, the tokenizer splits it into overlapping 6-mer tokens with a stride of 3 for the CDS classification task, while for TIS classification, we use a default stride of 1. After tokenization, each k-mer is transformed into a numerical representation using the pre-trained DNABERT model. Each k-mer is mapped to a fixed 768-dimensional vector, corresponding to the hidden size of the BERT [11] architecture. Additionally, we incorporate special tokens: `[CLS]` at the beginning and `[EOS]` at the end of each sequence. The `[CLS]` token provides an additional contextual representation for the sequence as a whole. As a result, for the CDS classification task, each sequence is represented as a matrix of size $(512, 768)$, while for the TIS classification, each sequence is represented as a matrix of size $(62, 768)$. This transfer learning enables the k-mer representations in each sequence to capture meaningful and contextually enriched information. Using DNABERT embeddings, we ensure that the tokenized sequences provide a robust foundation for downstream fine-tuning.

Model

Our model is based on the DNABERT framework, which extends the Bidirectional Encoder Representations from Transformers to genomic sequences. DNABERT follows the pre-training and fine-tuning paradigm, where the model first learns general DNA sequence representations and is later fine-tuned for specific downstream tasks. It retains the same transformer-based architecture as BERT, consisting of 12 self-attention layers, 768 hidden dimensions, and 12 attention heads, as shown in Figure 1.

Given a tokenized input sequence represented as k-mers, the model applies multi-head self-attention to learn contextual relationships across the sequence. The self-attention mechanism is formally defined as:

$$\text{MultiHead}(M) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{1}$$

where each attention head is computed as:

$$\text{head}_i = \text{softmax}\left(\frac{MW_i^Q(MW_i^K)^T}{\sqrt{d_k}}\right)MW_i^V \tag{2}$$

Here, M represents the input token embeddings, while W_i^Q, W_i^K, W_i^V are the learnable query, key, and value matrices for the i-th attention head. The self-attention scores determine the contextual relevance of each token concerning all others in the sequence. The final hidden representations from the transformer layers are then used for sequence-level or token-level classification tasks.

Fine-Tuning

For a given sequence, we prepend the classification token `[CLS]` to the tokenized input sequence and then generate its embedding representation using the DNABERT base model. Self-attention is applied, where the query and key matrices are both derived from the sequence itself. As illustrated in Figure

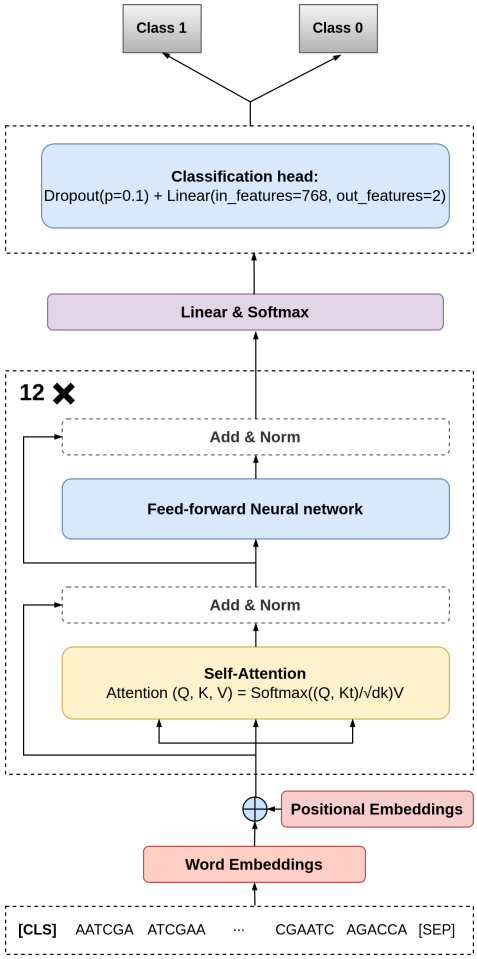

Fig. 1. Illustration of the transformer-based BERT architecture used for genomic sequence classification. The model consists of multiple self-attention layers that process k-mer tokenized DNA sequences, capturing both local and long-range dependencies. The input sequence is embedded and passed through 12 transformer layers, each containing 12 attention heads and hidden dimensions of size 768. The output representations are fine-tuned for specific downstream tasks such as coding sequence classification and translation initiation site prediction.

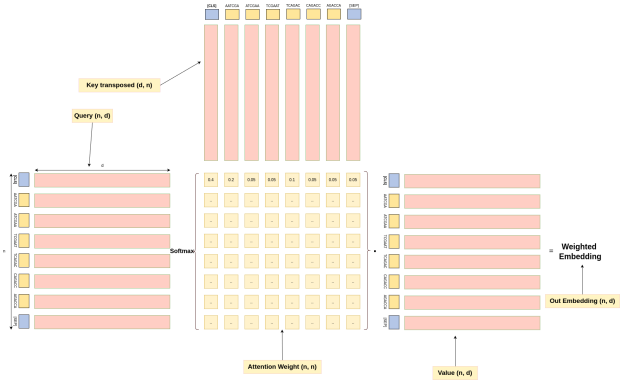

Fig. 2. Self attention illustration: The attention mechanism assigns dynamic importance to different k-mer tokens within the sequence by computing attention scores from query, key, and value matrices. The scores are normalized using a softmax function, and the resulting weighted sum refines the sequence embeddings.

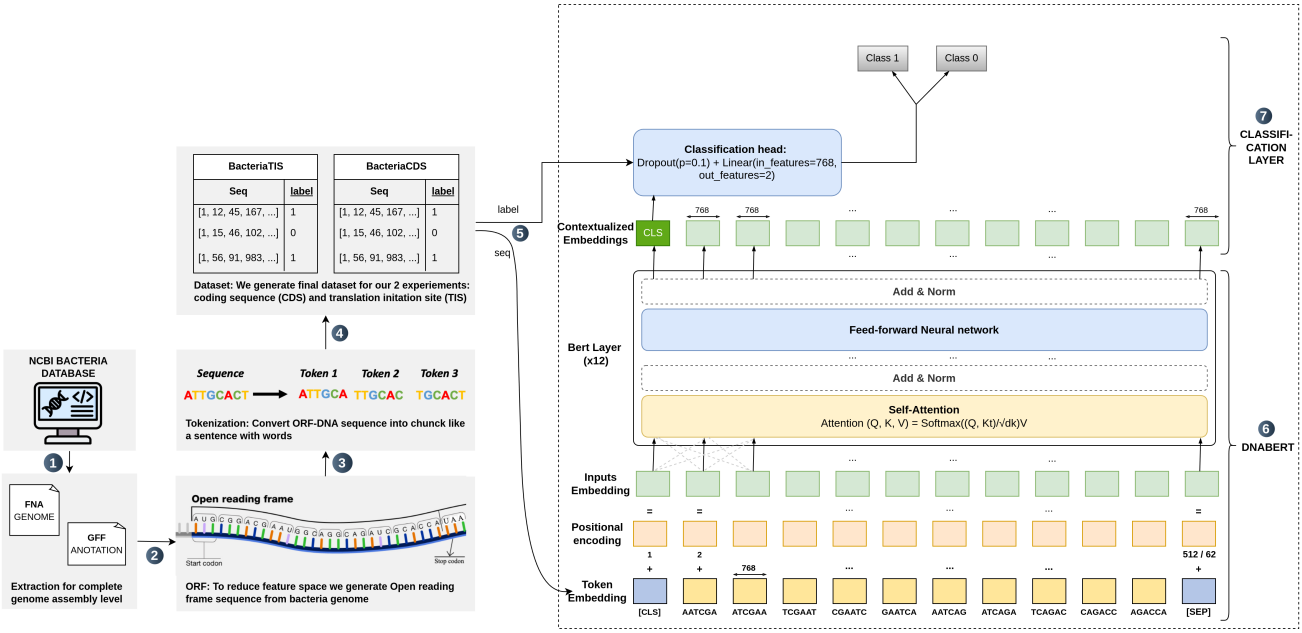

Fig. 3. TIS & CDS fine-tuning pipeline: The procces involves extracting Open Reading Frames from bacterial genome data, tokenizing sequences into k-mers, and training a transformer-based model to classify genomic regions. The self-attention mechanism enables the model to capture long-range dependencies within DNA sequences, enhancing prediction accuracy.

2, the attention mechanism computes attention weights using a dot product between the query and key matrices. A softmax function normalizes these weights, and the result is multiplied by the value matrix to adjust the original input embeddings. The re-weighted embeddings are then passed through feed-forward layers, culminating in a classification head appended to the DNABERT base architecture. The final classification decision is based on the embedding of the `[CLS]` token, which aggregates global sequence features.

Our fine-tuning approach focuses on two key classification tasks: identifying Coding Sequence regions from DNA sequences and detecting Translation Initiation Sites within CDS regions. Fine-tuning follows the standard BERT-based classification procedure, where the final hidden representation of the `[CLS]` token is passed through a fully connected layer for sequence-level classification. To enable this, DNABERT is extended by appending a classification head, which consists of:

- **Pooler Layer**: Aggregates information from the `[CLS]` token embedding to generate a fixed-size representation for the entire sequence.
- **Fully Connected Layers**: A feed-forward neural network with one or more layers that maps the pooled representation to the output classes.
- **Softmax Layer**: Converts the output logits into probability scores for classification.

The objective function for fine-tuning is the cross-entropy loss, defined as:

$$L = -\sum_{i=1}^{N} y_i' \log(y_i) \tag{3}$$

where y_i' represents the ground-truth labels, and y_i denotes the predicted probabilities for each of N classes. In our case for both experiments $N = 2$.

Experiment Setup

The entire workflow, from input tokenization and embedding generation to final classification, is summarized in the pipeline diagram (Figure 3). This diagram outlines all key components, including the DNABERT model, Transformer layers, and the classification head. Given the large-scale dataset, we implement an iterable dataset using PyTorch's data utilities to efficiently handle data loading and batch processing. The model is trained using the AdamW optimizer with a linear warm-up strategy. The learning rate is initialized at $3e-5$ and gradually decayed to zero over the training steps. The model undergoes fine-tuning on NVIDIA GPUs partitions. To ensure robust evaluation, we partition the dataset into training and testing subsets, where the test set contains only organisms absent from the training set. This experimental design ensures that the model's performance is assessed on previously unseen taxa, effectively evaluating its generalization ability.

1. **CDS Classification**
   The objective of this task is to classify coding sequences in genomic data. The CDS classification experiments were conducted on the Toubkal supercomputer. The training was performed on two NVIDIA A100 GPUs, each equipped with 80GB of memory. Distributed training was implemented across both GPUs with a batch size of 512, and the total training duration was approximately 37 hours for 2 epochs.
2. **TIS Classification**
   For Translation Initiation Site classification, the objective is to classify 60-bp sequences centered around the potential translation initiation site. This experiment was conducted at the College of Computing, Bioinformatics Lab using a single NVIDIA RTX A5000 GPU with 24GB of memory. The model was trained with a batch size of 768, and the training process completed in approximately 19 hours for 3 epochs.

Evaluation Metrics

We used a set of standard classification metrics to evaluate the performance of both the individual binary classifiers and the final classifier (via stacking or max-voting). These metrics provide a comprehensive overview of the model's ability to correctly classify protein transcription factor families. The following metrics were calculated:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

Post-processing

After TIS prediction, we applied a post-processing step to select the most likely translation initiation site for each predicted coding region. Among all candidate TIS positions within a given ORF group, the site with the highest predicted probability was retained. This decision was recorded using a binary flag (prediction_max_likelihood = 1), marking the optimal site per region. The final annotated results were then formatted into standard outputs, including CSV and GFF files, to ensure compatibility with common genome annotation tools and prepare the data for downstream analysis or visualization.

RESULTS

This section presents the results obtained from our experiments, structured into three key evaluation stages. First, we report the performance of our model during training and testing, assessing its effectiveness using standard evaluation metrics. Next, we evaluate the model's performance in a real-world setting by testing it on experimentally verified sequences, providing a practical assessment of its predictive accuracy on biological data validated in the laboratory. Finally, we compare our approach against state-of-the-art gene annotation tools, benchmarking its accuracy and efficiency. These evaluations provide a comprehensive understanding of the model's capabilities and its potential for practical applications in genome annotation.

Training performances

In this section, we present the results obtained from training our model across different experiments.

1. **CDS Classification**
   Figure 4 illustrates the training loss progression over multiple steps and the comparison of evaluation metrics across two fine-tuning epochs. Initially, the training loss decreases sharply as the model learns, stabilizing after several thousand iterations. However, sporadic spikes in the training loss were observed, potentially due to the composition of mini-batches with difficult examples. To investigate this, we conducted a reproducibility study by repeating the training with different random seeds and analyzing the loss dynamics. As detailed in Supplementary Section S3 and S4, the model showed consistent convergence patterns across seeds. We quantified the number of spikes and computed loss distribution statistics, confirming that these fluctuations are not indicative of instability but rather minor stochastic variations in training. Despite these fluctuations, the overall trend shows a consistent reduction in loss, affirming effective learning. After the second fine-tuning epoch, the evaluation metrics showed marginal improvements (Table 1).

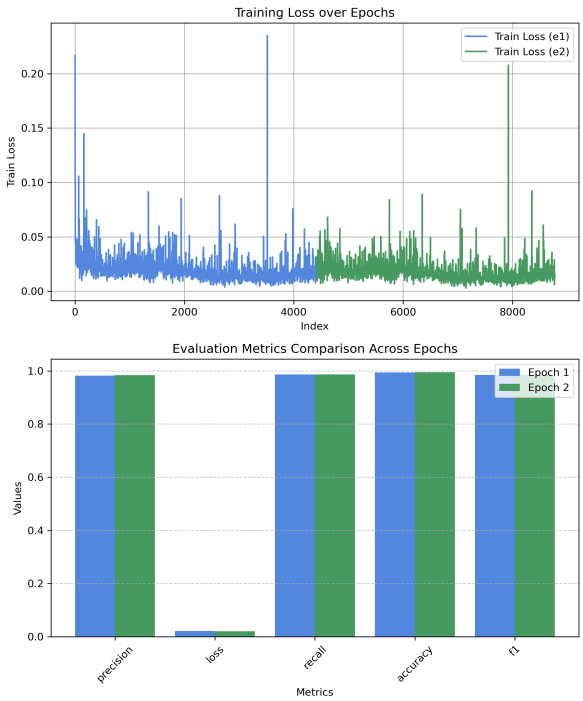

Fig. 4. Train and Evaluation Metrics Comparison Across Epochs for CDS classification experiment. Additional analysis of loss fluctuations is provided in Supplementary Section S3.

Table 1. Evaluation and test set performance metrics for CDS classification.

	Evaluation Set		Test Set
Metric	Epoch 1	Epoch 2	Final Score
loss	0.021233	0.020184	0.021132
Precision	0.981846	0.983868	0.983253
F1	0.984364	0.985109	0.984544
Accuracy	0.994265	0.994553	0.994364
Recall	0.986916	0.986357	0.985844

Note: The table presents evaluation metrics for both training epochs and the final test set. The model achieved a classification performance, with precision, recall, and accuracy exceeding 98%.

*Test set evaluation confirms the model's generalization ability with minimal loss and high precision.

The small gain in performance suggesting that continuing training further might yield diminishing returns

or risk overfitting, combined with the computational cost of additional training, led us to halt fine-tuning at epoch 2. Table 1 presents the evaluation metrics for both training epochs and the final test set. The model achieved high classification performance, with precision, recall, and accuracy exceeding 98%. A slight reduction in evaluation loss suggests stable and effective training across epochs. The final model was evaluated on an independent test set, maintaining high accuracy (99.43%) while preserving a low evaluation loss (0.0211). These results confirm the model's robustness and generalization capabilities.

2. **TIS Classification**

Figure 5 illustrates the training loss trend and evaluation metrics for the TIS classification experiment. Similar to CDS classification, the training loss shows a steep initial decline before stabilizing, with minor oscillations. The evaluation metrics in Table 2 show consistent performance improvements across epochs, with precision, recall, and F1-score increasing slightly over each iteration. However, as seen in the figure, these metrics plateau after the third epoch, showing no further meaningful gains—hence, training was stopped early at epoch 3 to avoid overfitting. The final test evaluation confirmed an accuracy (94.13%) and an evaluation loss (0.1546).

One should note that the TIS information annotation on NCBI and used to compute the above accuracy is done using automatic gene prediction tools which may not always reflect the ground truth. To provide a more reliable evaluation, we further benchmarked our tool against existing methods using all available experimentally verified TIS data for bacterial genomes.

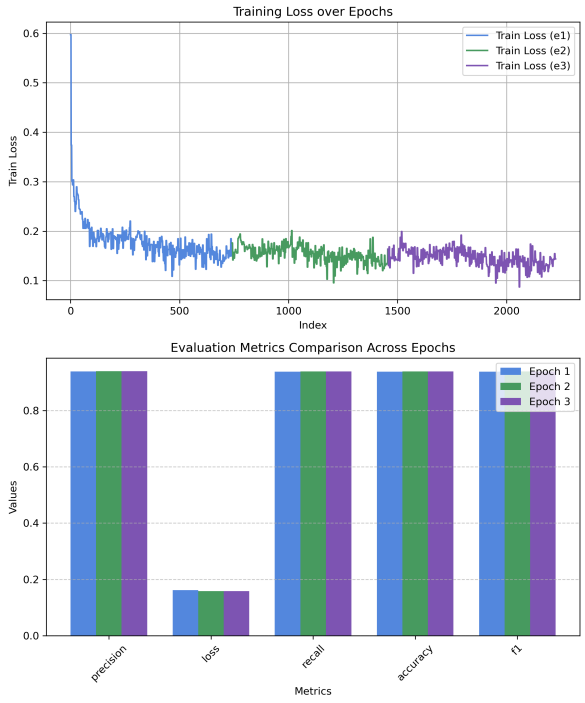

Fig. 5. Train and Evaluation Metrics Comparison Across Epochs for TIS classification experiment.

Table 2. Evaluation and test set performance metrics for TIS classification.

Metric	Evaluation Set			Test Set
	Epoch 1	Epoch 2	Epoch 3	Final Score
Loss	0.161598	0.158706	0.156721	0.154614
Precision	0.938289	0.939115	0.939882	0.941508
F1 Score	0.937736	0.938943	0.939723	0.941359
Accuracy	0.937755	0.938949	0.939728	0.941364
Recall	0.937754	0.938948	0.939728	0.941363

Note: The table presents evaluation metrics for both training epochs and the final test set. The model achieved a classification performance, with precision, recall, and accuracy exceeding 93%.

*Test set evaluation confirmed generalization ability with a final accuracy of 94.13%.

Evaluating Length Bias and Overfitting in CDS Classification

To ensure the robustness of our CDS classifier and reduce the risk of overfitting to simple features such as ORF length, we implemented a length-aware balancing strategy during training. In the CDS dataset, negative examples were downsampled based on ORF length to match the length distribution of positive (true CDS) examples. This approach was designed to discourage the model from relying on length as a shortcut and to promote learning of discriminative sequence features. To evaluate how well the model generalizes and whether prediction accuracy varies with ORF length, we conducted a length-stratified inference analysis. We used five verified genomes with verified CDS annotations from our benchmarking dataset. We extracted all candidate ORFs, predicted their coding status using the trained model, and compared predictions to ground-truth annotations. ORFs were then grouped into four bins by length, ranging from short (<300 bp) to very long (≥2000 bp). For each bin, we computed precision, recall, and accuracy, and also counted the number of ORFs falling into each length category.

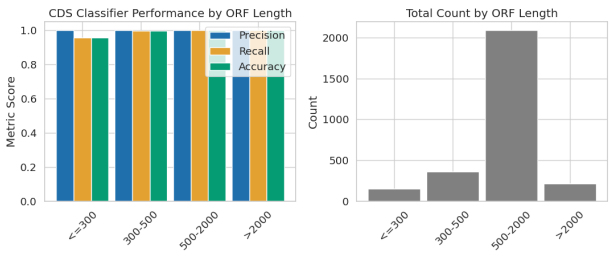

Fig. 6. Left: Aggregated CDS classifier performance metrics (precision, recall, accuracy) across ORF length bins using five verified genomes. Right: Distribution of ORFs by length. Performance is consistently high across bins, though shorter ORFs (<300 bp) remain less abundant.

As shown in Figure 6, the classifier achieves consistently high performance across all ORF lengths. However, we observe that shorter ORFs, particularly those under 300 bp, are still underrepresented in the test data, which may limit confidence in their performance metrics. Despite this limitation, the overall results provide strong evidence that the model does not rely predominantly on ORF length for classification. This affirms the importance and effectiveness of the length-balancing procedure used during training, which mitigates length-related

biases and enhances the biological relevance of learned sequence representations.

Performance Benchmarking on Experimentally Verified Genomes

The effectiveness of GeneLM was benchmarked using experimentally verified coding sequences. Benchmarking with real biological data is a standard practice for evaluating gene prediction methods, as seen in prior studie [13, 12, 14]. Building on this, we compared GeneLM against three widely used prokaryotic gene prediction tools: Prodigal [14], GeneMark-HMM [19] and Glimmer3 [10] as well as recent deep learning-based Translation Initiation Site predictors, including TITER [27], DeepGSR [16] and DeepTIS [26]. The evaluation was performed on five bacterial and archaeal genomes—*Escherichia coli*, *Halobacterium salinarum*, *Natronomonas pharaonis*, *Mycobacterium tuberculosis* and *Roseobacter denitrificans*—selected for their large sets of genes with experimentally verified TISs based on N-terminal peptide sequencing. These organisms (listed in Table 3) had the highest number of annotated TISs, comprising a test set of 2,841 verified genes [12].

Species	Clade	genomes in clade	verified genes
E. coli	*Enterobacterales*	6,311	769
H. salinarum	*Archaea*	1,125	530
N. pharaonis	*Archaea*	1,125	282
M. tuberculosis	*Actinobacteria*	8,097	701
R. denitrificans	*Alphaproteobacteria*	4,720	526

Table 3. Reference clades for the five query species and the sizes of the verified gene test sets. These species had the largest number of genes with starts verified by N-terminal sequencing (total of 2,841 genes) [12].

The reference sequences were obtained from public genome databases corresponding to each species verified annotation dataset [21], [28], [17], [6], [1], and were processed through the GeneLM pipeline as well as the comparative tools. All tools were executed using documented and reproducible command-line workflows reflecting best practices for each method (for more details on tool descriptions and execution details see Supplementary Section S5). Performance was measured in terms of correctly predicted TIS positions and the total number of coding sequences predicted (see Table 4).

`Prodigal, GeneMark, Glimmer:` Table 4 summarizes the performance across the five genomes. GeneLM consistently outperformed classical methods across all metrics. Specifically, it achieved the highest number of matched TIS (both 5' and 3' ends), with fewer missed predictions. For *E. coli K-12*, GeneLM missed only 25 TIS compared to 431 missed by Prodigal. GeneMark-HMM performed competitively, especially in high-GC genomes such as *M. tuberculosis* and *H. salinarum*, but still lagged behind GeneLM in 5' start precision. Glimmer3, while historically important, showed weaker performance in both configurations. Its scratch-based model performed worse than the iterated variant, confirming that upstream motif modeling is beneficial, but both approaches underperformed in comparison to GeneLM.

`TITER, DeepGSR, DeepTIS:` In addition to classical gene prediction tools, we also benchmarked GeneLM against recent deep learning approaches from the literature. Since these methods do not provide pretrained models suitable for direct inference, we retrained their architectures on our prokaryotic datasets for the specific task of Translation Initiation Site (TIS) prediction. Before evaluating them on experimentally verified genomes, we first assessed their performance on a balanced dataset using standard metrics: precision, recall, and F1-score. As shown in Supplementary Figure S5, TITER achieved higher recall (77.05%) but lower precision (66.03%), indicating a tendency to overpredict positive TIS sites. DeepGSR, on the other hand, offered more balanced performance (precision: 71.99%, recall: 73.11%) and a slightly higher F1-score. DeepGSR also required longer training (106.7 hours) compared to TITER (84.4 hours), likely due to its deeper architecture and larger parameter count. Subsequently, both models were benchmarked alongside GeneLM using experimentally verified TIS datasets across five bacterial genomes (Table 7). Although retrained on the same data for fairness, TITER and DeepGSR consistently underperformed relative to GeneLM. Notably, neither model was able to detect any verified TIS sites in *R. denitrificans*. DeepTIS could not be evaluated due to incomplete architectural and training information, as previously reported by [8].

`Evo2:` Recent advances in genomic foundation models have introduced Evo2 [7], a long-context DNA language model capable of modeling up to 1 million base pairs. Evo2 was pretrained autoregressively on 8.8 trillion nucleotides across domains of life and is designed for tasks such as sequence generation and zero-shot variant effect prediction. To explore Evo2's relevance to gene structure modeling, we used the Evo Mechanistic Interpretability Visualizer developed by the Arc Institute and Goodfire. This tool uses a sparse autoencoder trained to decode internal Evo2 activations into discrete, interpretable features. Trained on 100 bacterial genomes, the interpreter reveals features strongly aligned with known genomic concepts such as CDS, tRNA, and secondary structure motifs. We focused on Evo2 feature **f/13606**, which was identified as highly predictive of coding sequences (CDS). Using its exported activation data, we examined the whole *E. coli* K-12 genome sequence comparing GeneLM prediction to corresponding Evo2 activation values.

As shown in Figure 7 f/13606 activations consistently overlapped with GeneLM CDS predicted annotations. Notably, Evo2's unsupervised internal representation showed strong alignment with supervised outputs from GeneLM. These observations suggest that Evo2 feature activations may serve as an effective biological prior or quality-control signal for CDS annotation pipelines. However, we did not include Evo 2 in our benchmarks for several practical and methodological reasons. Evo 2's hardware requirements are prohibitive for most laboratories: the 7B model already requires a high-memory GPU ($\geq$40GB for full precision), while the 40B variant and context expansion stages rely on H100 GPUs with FP8 support and Vortex or BioNeMo stacks. In contrast, GeneLM operates efficiently on A100 and even A5000-class GPUs, with automatic batch scaling built into our pipeline. Finally, although Evo 2 offers significantly larger context windows, our current results with DNABERT indicate that most prokaryotic CDS regions and surrounding TIS signals fall well within the 512 bp limit. Thus, the benefit of the longer context may be marginal in the prokaryotic setting and comes at significant computational cost. While Evo2 offers compelling interpretability insights, we present it as a complementary tool for understanding genomic signals rather than a direct benchmark competitor, given its distinct design goals and computational constraints. Nevertheless, we acknowledge the potential of Evo 2 to enhance

Bacteria	GC	Verified TIS	GeneLM		Prodigal v3.0		GeneMark-HMM v2.8		Glimmer (scratch)		Glimmer (iterated)	
			Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end
E. coli	50.8	769	**744 (96.7%)**	**768 (99.9%)**	338 (44.0%)	345 (44.9%)	595 (77.4%)	759 (98.7%)	276 (35.9%)	366 (47.6%)	319 (41.5%)	369 (48.0%)
H. salinarum	65.7	530	438 (82.6%)	514 (97.0%)	243 (45.8%)	255 (48.1%)	**493 (93.0%)**	**530 (100%)**	220 (41.5%)	266 (50.2%)	220 (41.5%)	265 (50.0%)
M. tuberculosis	65.6	701	**626 (89.3%)**	**695 (99.1%)**	311 (44.4%)	342 (48.8%)	545 (77.7%)	694 (99.0%)	274 (39.1%)	353 (50.4%)	271 (38.7%)	352 (50.2%)
N. pharaonis	63.1	315	248 (78.7%)	302 (95.9%)	169 (53.7%)	176 (55.9%)	**302 (95.9%)**	**314 (99.7%)**	164 (52.1%)	178 (56.5%)	163 (51.7%)	178 (56.5%)
R. denitrificans	58.9	526	**492 (93.5%)**	**523 (99.4%)**	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	204 (38.8%)	273 (51.9%)	233 (44.3%)	275 (52.3%)
All Genomes	–	2841	**2548 (89.7%)**	**2802 (98.6%)**	1061 (37.3%)	1118 (39.4%)	1935 (68.1%)	2297 (80.9%)	1138 (40.1%)	1436 (50.5%)	1206 (42.4%)	1439 (50.7%)

Table 4. Comparison with Verified Annotations: Benchmarking of traditional TIS annotation tools across five experimentally verified bacterial genomes. Reported metrics include the number of matched TIS (5'+3' ends and 3' ends only). For each genome, the percentages of correctly predicted ends are shown in parentheses below the first two columns. The final row summarizes performance across the entire set of organisms.

Bacteria	GC	GenBank TIS	GeneLM		Prodigal v3.0		GeneMark-HMM v2.8		Glimmer (scratch)		Glimmer (iterated)	
			Matched (5'+3') end	Matched 3' end	Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end	Matched (5'+3')	Matched 3' end
E. coli	50.8	4140	**3767 (91.0%)**	**4033 (97.5%)**	1863 (45.0%)	1968 (47.6%)	3098 (74.8%)	3973 (96.0%)	1474 (35.6%)	2013 (48.6%)	1724 (41.6%)	2026 (49.0%)
H. salinarum	65.7	2749	1871 (68.1%)	2559 (93.1%)	1079 (39.3%)	1270 (46.2%)	**2174 (79.1%)**	**2590 (94.2%)**	886 (32.2%)	1244 (45.3%)	900 (32.7%)	1244 (45.3%)
M. tuberculosis	65.6	3906	**2664 (68.2%)**	**3709 (95.0%)**	1432 (36.7%)	1904 (48.7%)	2507 (64.2%)	3745 (95.9%)	1251 (32.0%)	1931 (49.4%)	1264 (32.4%)	1939 (49.6%)
N. pharaonis	63.1	2820	1978 (70.1%)	2671 (94.7%)	1313 (46.6%)	1461 (51.8%)	**2424 (86.0%)**	**2748 (97.4%)**	1186 (42.1%)	1483 (52.6%)	1217 (43.2%)	1484 (52.6%)
R. denitrificans	58.9	4057	**3278 (80.8%)**	**3927 (96.8%)**	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	1322 (32.6%)	2002 (49.3%)	1528 (37.7%)	2025 (49.9%)
All Genomes	–	17672	**13558 (76.7%)**	**16899 (95.6%)**	5687 (32.2%)	6603 (37.4%)	10203 (57.7%)	13056 (73.9%)	6119 (34.6%)	8673 (49.1%)	6633 (37.5%)	8718 (49.3%)

Table 5. Comparison with Genbank Annotations: Benchmark of gene-finding algorithms against GenBank annotations across five bacterial genomes. Each entry shows the number and percentage of predicted genes that matched GenBank annotations at the 3' end, as well as full matches (5'+3'). Although GenBank genes are not experimentally verified, this benchmark provides a broad overview of performance on complete genomes. This table is just meant to provide a snapshot of performance over entire genomes.

Bacteria	GeneLM	Prodigal v3.0	GeneMark-HMM v2.8	Glimmer (scratch)	Glimmer (iterated)
E. coli	4213	4347	4308	4397	4478
H. salinarum	2659	2851	2762	2717	2762
M. tuberculosis	3853	4204	4029	4240	4349
N. pharaonis	2737	2873	2826	2878	2894
R. denitrificans	4006	4120	4104	4260	4345
All Genomes	17468	18395	18029	18492	18828

Table 6. Total Predicted CDS: Comparison of total predicted coding sequences (CDS) across five gene prediction tools for five experimentally verified bacterial genomes. The final row shows the total CDS predictions aggregated across all genomes.

Species	GC	Verified TIS	GeneLM (5' end)	TITER (5' end)	DeepGSR (5' end)
E. coli	50.8	769	**744 (96.7%)**	662 (86.1%)	647 (84.1%)
H. salinarum	65.7	530	**438 (82.6%)**	391 (73.8%)	395 (74.5%)
M. tuberculosis	65.6	701	**626 (89.3%)**	493 (70.3%)	459 (65.5%)
N. pharaonis	63.1	315	**248 (78.7%)**	220 (69.8%)	208 (66.0%)
R. denitrificans	58.9	526	**492 (93.5%)**	0 (0.0%)	0 (0.0%)
All Genomes	–	2841	**2548 (89.7%)**	1766 (62.2%)	1709 (60.2%)

Table 7. Comparison with Deep Learning Approach on TIS Taks: Translation Initiation Site prediction performance of deep learning models across five experimentally verified bacterial genomes. Values indicate the number of correctly predicted TIS at the 5' end, with corresponding percentages shown in parentheses. GeneLM achieves the highest performance across all organisms.

Fig. 7. Evo2 Activation vs. GeneLM CDS Predictions on the *E. coli* Genome. Multi-scale visualization of Evo2 feature activation (`f/13606`) alongside GeneLM-predicted CDS regions. Top panel: Full genome view showing consistent alignment between Evo2 activations and GeneLM CDS predictions across the *E. coli* K-12 genome. Middle panel: 1 Mbp region zoomed to highlight local activation density and fine-grained GeneLM boundaries. Bottom panel: Zoom into a 15 kbp region illustrating the near-basepair alignment between Evo2 feature activations and GeneLM-predicted ORFs. These patterns suggest that Evo2 captures biologically relevant signals for coding regions, closely matching the supervised outputs of GeneLM. Visualization adapted using IGV.js [24], [20].

CDS classification and plan to explore its embeddings and context capabilities in future work for hybrid annotation workflows.

Explaining TIS Predictions Through Attention-Based Motif Visualization & Attention-Guided Sequence Disruption

The task of Translation Initiation Site classification involves categorizing 60-nucleotide windows surrounding the TIS site. To understand the model's decision-making process, we analyze how attention mechanisms contribute to classification. As shown in Supplementary Figure S6 and S7, the model captures distinct patterns through its attention heads. For example, as observed in layer 1, the 11th head focuses on a region approximately 30 base pairs upstream of the TIS, which may indicate the presence of a promoter site. In contrast, layer 2 exhibits a more selective focus on upstream regions, and a similar pattern is noticeable in layer 9. To derive more generalized insights, instead of analyzing isolated sequences, we extend our visualization across all sequences in our verified test set, which consists of five bacterial species. We focus on layer 11, the final attention layer, as it aggregates all prior learned patterns and directly influences the classification head, playing a crucial role in the final decision. For true TIS sites, we compute attention weights using our TIS prediction model, yielding fixed-size tensors of 11 heads with dimensions 57×57. The mean attention weights are calculated over all 11 heads per sequence and then averaged across all TIS instances.

Figure 8 presents the attention weight landscape for each bacterial species. The heatmaps labeled **(a)**, **(b)**, **(c)**, **(d)**, and **(e)** correspond to *Escherichia coli*, *Halobacterium salinarum*, *Mycobacterium tuberculosis*, *Natronomonas pharaonis*, and *Roseobacter denitrificans*, respectively. The red box highlights the immediate region surrounding the TIS, while the blue box reveals an intriguing pattern: it marks sequence regions that receive high attention from the classification (CLS) token. These upstream regions may contain potential promoter elements, as the CLS token is responsible for sequence-level classification. This pattern is systematically observed across all bacterial species and aligns with the expected biological significance of promoter regions.

To further assess the biological relevance of the model's learned attention patterns, we conducted an attention-guided disruption experiment. The aim was to test the sensitivity of TIS prediction scores to perturbations in sequence regions deemed important by the model's attention weights. Specifically, for each verified TIS-containing sequence, we

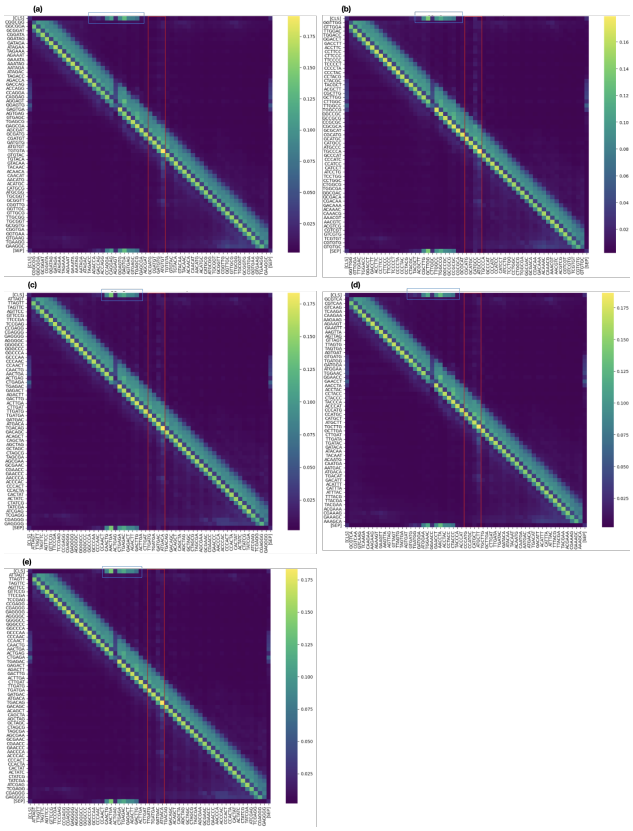

Fig. 8. Mean attention weight distribution across different bacterial species. Each heatmap represents a verified bacterial species, where the mean attention weights are computed for sequences labeled as true TIS. This visualization highlights the regions where the classification token `[CLS]` focuses the most when making the final decision.

performed systematic replacements in either high-attention or low-attention regions. The disruption ratio ranged from 0.0 (no modification) to 1.0 (full substitution of the region), and at each ratio, we recorded the model's TIS prediction probabilities. Two disruption modes were applied: **High Attention Disruption**, where substitutions targeted positions with the highest attention scores; and **Low Attention Disruption**, where changes were applied to the least-attended positions.

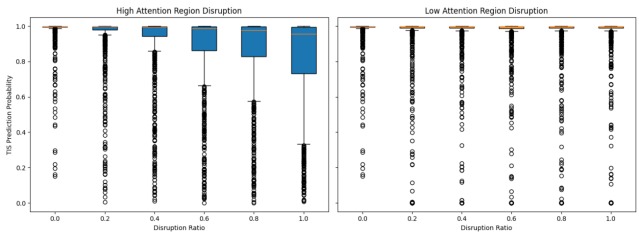

Fig. 9. Effect of attention-guided sequence disruption on TIS prediction probabilities. Left: Boxplot of predicted probabilities under increasing disruption of high-attention regions. Right: Boxplot for low-attention region disruption. Each box represents the distribution of predicted TIS probabilities at a given disruption ratio across all sequences.

As illustrated in Figure 9, disrupting high-attention regions significantly reduces TIS prediction probabilities as the disruption ratio increases. This suggests that the model heavily relies on the sequence information within these regions when making classification decisions. In contrast, disrupting low-attention regions had little to no impact on prediction scores, with most outputs remaining close to the original probability distribution. These findings validate the model's ability to attend to biologically informative regions and further support the interpretability of attention mechanisms in genomic sequence classification tasks.

GeneLM Inference Scalability and GPU Resource Analysis

We evaluated the scalability and practical feasibility of GeneLM by benchmarking its inference performance on three bacterial genomes (2.67M to 4.64M bp) across three GPU models with varying memory capacities: NVIDIA A100 (80GB), RTX A5000 (24GB), and RTX A2000 (12GB). Using identical model configurations, we recorded the batch size and total inference time for each GPU-genome combination (Table 8). To maximize efficiency, the GeneLM pipeline automatically scales the batch size based on available GPU RAM, making it highly adaptable across different hardware settings

Genome	GPU Name	Memory (GB)	Batch Size	Genome Size (bp)	Inference Time (min)
bacteria-1	**A100**	79.15	5375	4,641,652	21.12
bacteria-2	**A100**	79.15	5375	2,668,776	9.75
bacteria-3	**A100**	79.15	5375	4,411,532	21.73
bacteria-1	**RTX A5000**	23.66	1556	4,641,652	33.08
bacteria-2	**RTX A5000**	23.66	1556	2,668,776	15.60
bacteria-3	**RTX A5000**	23.66	1556	4,411,532	34.80
bacteria-1	**RTX A2000**	11.65	710	4,641,652	101.76
bacteria-2	**RTX A2000**	11.65	710	2,668,776	47.74
bacteria-3	**RTX A2000**	11.65	710	4,411,532	106.20

Table 8. Inference benchmarking of GeneLM across different GPU models and bacterial genomes. The table reports GPU memory, batch size, runtime in minutes, and genome size, providing insight into resource requirements and performance scaling.

As shown in Table 8, GeneLM's inference time scales with both genome size and GPU memory. On the A100, batch sizes exceed 5000, enabling rapid annotation (under 22 minutes for genomes over 4.5M bp). The A5000 performs reasonably well (≈33–35 minutes), while the A2000, constrained by smaller memory and lower batch sizes (710), shows much longer runtimes (up to 106 minutes). To better compare GPU efficiency across genomes of different lengths, we normalized the inference time by genome size (in Mbp). This yields an average inference speed of approximately 4.5 minutes per Mbp for the A100, compared to 7.5 minutes per Mbp on the A5000 and 22+ minutes per Mbp on the A2000. These values are derived by dividing the total inference time by the genome length in millions of base pairs (e.g., 21.12 min / 4.64 Mbp ≈ 4.55 min/Mbp).

This performance gap reflects the memory-bound nature of transformer-based inference: larger batch sizes reduce padding and memory transfers, increasing throughput. Despite this, the model remains usable on mid-range GPUs like the A5000, making it accessible for many research labs.

In future work, inference speed could be further improved by applying techniques such as mixed-precision inference,

quantization, or model distillation. These enhancements would help adapt GeneLM for environments with limited computational resources, facilitating broader adoption.

GeneLM Web Tool: Web Application and API

To make GeneLM accessible to a broader community of researchers and bioinformaticians, we developed a web-based genome annotation interface and a RESTful API. These tools serve as practical applications of the model, allowing users to annotate bacterial genomes using GeneLM without requiring deep computational expertise. The tool is publicly available on GitHub at github.com/Bioinformatics-UM6P/GeneLM.

GeneLM web tool supports two modes of input: direct input where users can paste a genome sequence into the provided text area or file upload where users can upload a FASTA file for processing. Once the input is provided, users can specify the desired output format, selecting either GFF or CSV. Upon submission, the system processes the annotation and generates structured output files. The user-friendly interface ensures accessibility for researchers and bioinformaticians. Supplementary Figures S8 and S9 show a snapshot of the interface and an example of the downloadable annotated output, respectively.

Beyond the graphical interface, we developed a RESTful API to enable programmatic access for flexible genome annotation. The API allows users to submit annotation tasks asynchronously, queue multiple annotation jobs efficiently, track annotation progress, retrieve annotated results and cancel annotation tasks if necessary. The API is well-documented, providing multiple endpoints for annotation submission, result retrieval, and task management. This ensures seamless integration into bioinformatics pipelines and allows users to perform genome annotation directly through Python scripts or other computational workflows. This integrated web-based and API-driven annotation pipeline provides an efficient, scalable, and user-friendly solution for genome annotation, bridging the gap between machine learning-based gene prediction and real-world biological research.

DISCUSSION AND CONCLUSION

This work demonstrates the potential of transformer-based Genomic Language Models (gLMs) for accurate and interpretable bacterial gene annotation. Built upon the DNABERT architecture, GeneLM introduces a biologically structured two-stage classification pipeline that separates the tasks of Coding Sequence (CDS) identification and Translation Initiation Site (TIS) refinement. This modular design aligns with biological gene structure and contributes to enhanced prediction precision across diverse bacterial species.

A key strength of GeneLM lies in its ability to generalize across a wide range of organisms while maintaining high accuracy and recall. This is supported by a carefully curated and balanced dataset, including length-stratified sampling to mitigate ORF length bias. Qualitative and quantitative analyses of self-attention maps further revealed that the model consistently focuses on biologically meaningful upstream regions near TIS sites, suggesting sensitivity to promoter-like motifs. These insights were reinforced by attention-guided sequence disruption experiments, which demonstrated that perturbing high-attention regions significantly alters the model's predictions.

Although the model is initially pretrained on human genomic sequences to leverage large-scale self-supervised learning, we acknowledge the substantial evolutionary and organizational differences between eukaryotic and prokaryotic genomes. To address this domain gap, GeneLM is fine-tuned extensively on a large and diverse bacterial dataset, enabling adaptation to prokaryotic-specific features. The strong performance observed across multiple bacterial clades—including those with high-GC content—and the benchmarking against experimentally verified datasets provide evidence that this adaptation is effective in practice.

In benchmarking experiments using experimentally verified genomes, GeneLM achieved state-of-the-art performance in recovering annotated TIS positions and reducing false positives compared to classical gene prediction tools like Prodigal, GeneMark-HMM, and Glimmer, as well as recent deep learning approaches. Additionally, our evaluation of inference scalability across three GPU tiers shows that GeneLM adapts batch size based on available memory, enabling efficient deployment even on mid-range hardware. To further increase accessibility, we release GeneLM as an open-source web tool and API to facilitate its integration into existing annotation workflows. While transformer-based models impose higher computational demands than heuristic methods, future directions include mixed-precision inference, quantization, and model distillation to reduce resource usage.

In summary, GeneLM provides a reproducible and interpretable framework for bacterial gene boundary identification. Its contributions span task formulation, dataset design, biological insight, and deployment readiness—offering a solid foundation for further advancements in automated genome annotation and potential extensions toward non-coding and regulatory region discovery.

Code and data availability

The current implementation of our work and the trained models and their weights are available at: github.com/Bioinformatics-UM6P/GeneLM.

Author contributions statement

A.E. conceived the experiment(s), G.A. conducted the experiment(s), A.E. and G.A. analyzed the results. A.E validated the results. G.A. wrote the manuscript. All authors reviewed the manuscript.

Acknowledgments

The authors express their gratitude to the College of Computing at Mohamed VI Polytechnic University for providing access to the supercomputing resources (cc.um6p.ma/toubkal-super-computer) used in conducting the research presented in this paper.

Key Points

- A BERT-based genomic language model is developed to identify Coding Sequence regions and classify Translation Initiation Sites in Prokaryotes genomes.
- The model is initially trained using self-supervised learning on human genomic datasets to learn DNA chunk representations. It is then adapted for gene annotation

through a two-stage pipeline: first, a model classifies ORFs into CDS and non-CDS regions, followed by a refinement stage where a second model identifies true TIS sites.
- Comparative evaluations are conducted against traditional gene prediction tools, assessing improvements in precision, recall, and scalability. A case study on verified bacterial genomes is performed to demonstrate the model's applicability and robustness in real-world genomic annotation tasks.

Biographical Note

- **Genereux Akotenou** is a Masters student at the College of Computing, at Mohammed VI Polytechnic University, Morocco. His research interests are artificial intelligence and data science.
- **Achraf El Allali** is an associate professor and the head of the Bioinformatics Laboratory at the College of Computing at Mohammed VI Polytechnic University, Morocco. His research interests are computational biology, artificial intelligence and bioinformatics.

References

1. M. Aivaliotis, K. Gevaert, M. Falb, A. Tebbe, K. Konstantinidis, B. Bisle, C. Klein, L. Martens, A. Staes, S. Timmermann, and et al. Large-scale identification of n-terminal peptides in the halophilic archaea halobacterium salinarum and natronomonas pharaonis. *Journal of Proteome Research*, 6(5):2195–2204, 2007.
2. Amani Al-Ajlan and Achraf El Allali. Cnn-mgp: Convolutional neural networks for metagenomics gene prediction. *Interdisciplinary Sciences: Computational Life Sciences*, 11(4):628–635, December 2019.
3. Achraf El Allali and John R. Rose. Mgc: a metagenomic gene caller. *BMC Bioinformatics*, 14(9):S6, June 2013.
4. Gonzalo Benegas, Chengzhong Ye, Carlos Albors, Jianan Canal Li, and Yun S. Song. Genomic language models: Opportunities and challenges, 2024.
5. J. Besemer, A. Lomsadze, and M. Borodovsky. Genemarks: a self-training method for prediction of gene starts in microbial genomes. implications for finding sequence motifs in regulatory regions. *Nucleic Acids Research*, 29(12):2607–2618, 2001.
6. C. Bland, E. M. Hartmann, J. A. Christie-Oleza, B. Fernandez, and J. Armengaud. N-terminal-oriented proteogenomics of the marine bacterium roseobacter denitrificans och114 using tmpp labeling and diagonal chromatography. *Molecular & Cellular Proteomics*, 13(5):1369–1381, 2014.
7. Garyk Brixi, Matthew G Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A Gonzalez, Samuel H King, David B Li, Aditi T Merchant, Mohsen Naghipourfar, Eric Nguyen, Chiara Ricci-Tam, David W Romero, Gwanggyu Sun, Ali Taghibakshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K Wang, Etowah Adams, Stephen A Baccus, Steven Dillmann, Stefano Ermon, Daniel Guo, Rajesh Ilango, Ken Janik, Amy X Lu, Reshma Mehta, Mohammad R.K. Mofrad, Madelena Y Ng, Jaspreet Pannu, Christopher Re, Jonathan C Schmok, John St. John, Jeremy Sullivan, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Tom McGrath, Kimberly Powell, Dave P. Burke, Hani Goodarzi, Patrick D Hsu, and Brian Hie. Genome modeling and design across all domains of life with evo 2. *bioRxiv*, 2025.
8. Jim Clauwaert, Zahra McVey, Ramneek Gupta, and Gerben Menschaert. Tis transformer: remapping the human proteome using deep learning. *NAR Genomics and Bioinformatics*, 5(1):lqad021, 2023.
9. A. L. Delcher, D. Harmon, S. Kasif, O. White, and S. L. Salzberg. Improved microbial gene identification with glimmer. *Nucleic Acids Research*, 27(23):4636–4641, 1999.
10. Arthur L. Delcher, Kirsten A. Bratke, Edwin C. Powers, and Steven L. Salzberg. Identifying bacterial genes and endosymbiont dna with glimmer. *Bioinformatics*, 23(6):673–679, 2007.
11. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
12. Karl Gemayel, Alexandre Lomsadze, and Mark Borodovsky. Startlink and startlink+: Prediction of gene starts in prokaryotic genomes. *Frontiers in Bioinformatics*, 1, 2021.
13. Gang-Qing Hu, Xiaobin Zheng, Huai-Qiu Zhu, and Zhen-Su She. Prediction of translation initiation site for microbial genomes with tritisa. *Bioinformatics*, 25(1):123–125, 11 2008.
14. Doug Hyatt, Gwo-Liang Chen, Philip F. LoCascio, Miriam L. Land, Frank W. Larimer, and Loren J. Hauser. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, 11(1):119, 2010.
15. Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, 02 2021.
16. Manal Kalkatawi, Arturo Magana-Mora, Boris Jankovic, and Vladimir B. Bajic. Deepgsr: an optimized deep-learning structure for the recognition of genomic signals and regions. *Bioinformatics*, 35(7):1125–1132, 2019.
17. J. M. Lew, A. Kapopoulou, L. M. Jones, and S. T. Cole. Tuberculist–10 years after. *Tuberculosis*, 91(1):1–7, 2011.
18. Yuanhao Li, Bo Jiang, and Weijun Dai. A large-scale whole-genome sequencing analysis reveals false positives of bacterial essential genes. *Applied Microbiology and Biotechnology*, 106(1):341–347, 2022.
19. Alexander V. Lukashin and Mark Borodovsky. Genemark.hmm: New solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, 1998.
20. James T. Robinson, Helga Thorvaldsdóttir, Douglass Turner, and Jill P. Mesirov. igv.js: an embeddable javascript implementation of the integrative genomics viewer (igv). *Bioinformatics*, 39(1), 2023.
21. K. E. Rudd. Ecogene: a genome sequence database for escherichia coli k-12. *Nucleic Acids Research*, 28(1):60–64, 2000.
22. Urminder Singh and Eve Syrkin Wurtele. orfipy: a fast and flexible tool for extracting orfs. *Bioinformatics*, 37(18):3019–3020, 02 2021.
23. Maike Tech, Nico Pfeifer, Burkhard Morgenstern, and Peter Meinicke. Tico: a tool for improving predictions of prokaryotic translation initiation sites. *Bioinformatics*, 21(17):3568–3569, 2005. Epub 2005 Jun 30.

24. Helga Thorvaldsdóttir, James T. Robinson, and Jill P. Mesirov. Integrative genomics viewer (igv): high-performance genomics data visualization and exploration. *Briefings in Bioinformatics*, 14(2):178–192, 2013.
25. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.
26. Chao Wei, Junying Zhang, and Yuan Xiguo. Deeptis: Improved translation initiation site prediction in genomic sequence via a two-stage deep learning model. *Digital Signal Processing*, 117:103202, 2021.
27. Sai Zhang, Hailin Hu, Tao Jiang, Lei Zhang, and Jianyang Zeng. Titer: predicting translation initiation sites by deep learning. *Bioinformatics*, 34(4):516–524, 2018.
28. J. Zhou and K. E. Rudd. Ecogene 3.0. *Nucleic Acids Research*, 41(D1):D613–D624, 2013.

Figure Legends

Figure 1: Illustration of the transformer-based BERT architecture used for genomic sequence classification. The model consists of multiple self-attention layers that process k-mer tokenized DNA sequences, capturing both local and long-range dependencies. The input sequence is embedded and passed through 12 transformer layers, each containing 12 attention heads and hidden dimensions of size 768. The output representations are fine-tuned for specific downstream tasks such as coding sequence classification and translation initiation site prediction.
Alt Text: Diagram of a transformer-based BERT model for genomic sequence classification. The model includes 12 layers with self-attention heads, taking embedded DNA k-mers as input and outputting features for classification tasks.

Figure 2: Self-attention illustration: The attention mechanism assigns dynamic importance to different k-mer tokens within the sequence by computing attention scores from query, key, and value matrices. The scores are normalized using a softmax function, and the resulting weighted sum refines the sequence embeddings.
Alt Text: Schematic of self-attention mechanism highlighting how attention scores are computed using query, key, and value matrices. It shows how tokens are weighted and combined to form new embeddings.

Figure 3: TIS & CDS fine-tuning pipeline: The process involves extracting Open Reading Frames from bacterial genome data, tokenizing sequences into k-mers, and training a transformer-based model to classify genomic regions. The self-attention mechanism enables the model to capture long-range dependencies within DNA sequences, enhancing prediction accuracy.
Alt Text: Workflow diagram illustrating the pipeline from ORF extraction to classification using DNABERT, showing steps such as k-mer tokenization, transformer layers, and final classification outputs for CDS and TIS tasks.

Figure 4: Train and Evaluation Metrics Comparison Across Epochs for CDS classification experiment. Additional analysis of loss fluctuations is provided in Supplementary Section S3.
Alt Text: Line plots comparing training loss and evaluation metrics over two epochs for CDS classification, showing decreasing loss and improving performance.

Figure 5: Train and Evaluation Metrics Comparison Across Epochs for TIS classification experiment.
Alt Text: Line plots showing training loss and evaluation metrics (precision, recall, accuracy, F1) for TIS classification across three training epochs, with trends stabilizing by epoch 3.

Figure 6: Left: Aggregated CDS classifier performance metrics (precision, recall, accuracy) across ORF length bins using five verified genomes. Right: Distribution of ORFs by length. Performance is consistently high across bins, though shorter ORFs (¡300 bp) remain less abundant.
Alt Text: Bar charts showing CDS classifier performance (left) across ORF length bins and corresponding ORF count distribution (right), indicating robust performance across ORF lengths.

Figure 7: Evo2 Activation vs. GeneLM CDS Predictions on the *E. coli* Genome. Multi-scale visualization of Evo2 feature activation (f/13606) alongside GeneLM-predicted CDS regions. Top panel: Full genome view showing consistent alignment between Evo2 activations and GeneLM CDS predictions. Middle panel: 1 Mbp region zoomed to highlight local activation density and fine-grained GeneLM boundaries. Bottom panel: Zoom into a 15 kbp region illustrating the near-basepair alignment between Evo2 feature activations and GeneLM-predicted ORFs. These patterns suggest that Evo2 captures biologically relevant signals for coding regions, closely matching the supervised outputs of GeneLM.
Alt Text: Three-panel genomic visualization showing alignment between Evo2 feature activations and GeneLM CDS predictions on the *E. coli* genome, from genome-wide to basepair-level detail.

Figure 8: Mean attention weight distribution across different bacterial species. Each heatmap represents a verified bacterial species, where the mean attention weights are computed for sequences labeled as true TIS. This visualization highlights the regions where the classification token [CLS] focuses the most when making the final decision.
Alt Text: Heatmaps showing attention distributions from the final transformer layer for five bacterial species, emphasizing high-attention regions upstream of TIS positions.

Figure 9: Effect of attention-guided sequence disruption on TIS prediction probabilities. Left: Boxplot of predicted probabilities under increasing disruption of high-attention regions. Right: Boxplot for low-attention region disruption. Each box represents the distribution of predicted TIS probabilities at a given disruption ratio across all sequences.
Alt Text: Side-by-side boxplots showing how disrupting high-attention vs. low-attention regions affects TIS prediction scores, with high-attention disruption causing a significant decline in predicted probabilities.