Bioinformatics, 2025, 4–18

doi: 10.1093/bioinformatics/xxxxx

Advance Access Publication Date: 18 April 2025

Machine Learning

Machine Learning / Deep Learning

Fine-Tuning DNABERT for Bacterial Gene Identification and Classification

Justin Morera[1]

[1]University of Central Florida Department of Computer Science, mustinjorera@gmail.com

Associate Editor: Dr. Haiyan Hu. Ph.D.

University of Central Florida Department of Computer Science, haihu@cs.ucf.edu

Received on 4/20/2025

Abstract

Motivation: The goal of this project is to gain a better understanding of the capabilities of popular transformer models, DNABERT, in this case, at identifying and classifying genes on bacterial genome sequences. This involved comparing the pretrained, high-performing base model on bacterial genome sequences, an input on which it was not originally trained, to two fine-tuned versions of the same model trained for identification and classification on bacterial sequences, respectively.

Results: The fine-tuned models were able to learn how to identify significant patterns in both gene presence identification and gene type classification. The gene identification model was able to achieve an accuracy score of 0.94 and precision and recall scores of 0.97 each on 7'143'946 6-mers taken from ten representative genomes. On the same input, the gene classification model was able to achieve an F1-score of 0.97 for CDS-labeled genes, 0.92 for rRNA genes, and 0.32 for tRNA genes while all other gene types were not recognized by the model, each receiving an F1-score of 0. These results are expected when considering that the ten reference genomes used as input had a gene type distribution of 97.87% CDS, 1.41% rRNA, and 0.27% tRNA, meaning the other five gene types combined occupied only 0.45% of the total input. The most limiting factor to the models' performance was the size and distribution of the data, which was in turn limited by the local processing power available to the researcher.

Contact: mustinjorera@gmail.com

Supplementary information: Supplementary data are available in the official GitHub repository.

1 Introduction

Transformer models are one of the most influential innovations in contemporary deep learning, with applications across countless fields, especially in bioinformatics. Specifically, transformers excel in identifying patterns among complex, unstructured data, which is certainly the case with genomic data, where a multitude of factors can affect the biological output of a sequence segment made up of only 4 possible nucleotides (Vaswani et al., 2017). Transformers are able to discern many of these patterns by creating high-dimensional embedding vectors while simultaneously paying attention to both the order of tokens as well as to their own evolving representation of tokens as inputs are processed through deeper layers.

DNABERT is a specific implementation of the BERT transformer model that shines in the world of bioinformatics because of its specialized training on genetic sequences (Ji et al., 2021). It is trained to directly identify patterns among DNA strands, generally in the form of k-mers, and generally up to 512 k-mers per sequence (Ji et al., 2021). While DNABERT excels at identifying patterns on human genome sequences, and with DNABERT-2, this functionality has been extended to some other eukaryotic groups, there remains a lack of coverage for prokaryotic sequences, particularly the most prevalent group, bacteria James et al. 2025).

2 Methods

This project was completed in four parts. First, genomic data was retrieved from NCBI's many databases using various methods, explained

below. Second, the downloaded files were parsed, and relevant information was extracted and stored in a way amicable to downstream tasks. Third, the base DNABERT model was evaluated on its performance on gene identification and classification using the same embeddings created from the extracted input without any fine-tuning or retraining. And finally, two models extending the DNABERT architecture were customized and fine-tuned on slightly more streamlined input data from the same downloaded dataset; their performance on their respective tasks was compared to that of the base model's.

2.1 Data Retrieval via Entrez API and Direct Download

Multiple methods for data attainment were implemented in this project as a way of examining pipelining methods for bioinformatics analysis. NCBI's Genome database was manually filtered and perused to download a quantity of RefSeq-only bacterial reference genome GBFF files, including both the full genomic sequences and their corresponding gene annotations. Ultimately, this was the main method used for data collection; however, this method is not nearly as feasible for large-scale or repeated use due to manual filtering and download sizes, which justifies exploration into the other methods.

Other methods developed were NCBI's Entrez API eSearch tool, which is functional in the Jupyter notebook and allows the user to query NCBI's Assembly database for a specific number of bacterial reference genomes with RefSeq data available; note that the project can work with raw GenBank records, but these were avoided due to inconsistencies and a lack of curation compared to RefSeq genome files. The eSearch returns a list of assembly IDs which are in turn passed to another function which uses the Entrez API eSummary tool to extract the RefSeq GBFF file location for the organism with that ID. The link is then retrieved via a standard Python URL retrieval function which downloads the file as a compressed .gz file.

A third method was unsuccessfully implemented, which uses the Entrez API eSummary tool to find the Accession ID of the organism and passes that ID to the eSearch tool to query the NCBI Nucleotide database for the sequence and annotations remotely, to avoid downloading a GBFF file entirely. The issue is that the formatting is different than the previous two methods in terms of both the databases used and the data structure of the sequences and their annotations. Time constraints prevented deciphering the structure of the required data and so the fetched information was not able to be easily passed into the models for input.

The first method was used to download 52 reference genomes from the Genome database directly while the second method was also used to download an additional ten genomes to be used as inputs throughout the experiment.

2.2 Data Extraction and Preprocessing

For the directly downloaded files, regardless of origin, any compressed .gz files were first uncompressed into .gbff format, then each GBFF file was parsed algorithmically using the BioPython package which allows direct NCBI genomic file parsing. Each sequence was extracted and stored as a dictionary along with its length, the accession number, the organism name, subspecies, molecule type, strain, and serovar. The dictionary also includes the full taxonomy from the Domain to Species level, as well as the sequences accompanying description, the genome type ("chromo" or "plasmid"), the recorded host organism, if present, and the full list of gene annotations.

Each gene list represents each gene as a dictionary storing its type, name, start base pair, end base pair, total length, calculated GC content and intergenic distance. Additionally, it stores fields for each possible product the gene could produce, depending on its type. These fields include element types for mobile elements, bound moieties for protein-binding genes, protein products with their amino acid sequence for CDS genes, regulatory classes for regulatory genes, and classes and products for all RNA types. Most of this data is reserved for expanded downstream analyses as the trained models only used the sequences as inputs and the gene types and start and end indices as labels.

The expansive list of sequence dictionaries included data for 113 sequences obtained from the 62 downloaded files. Ten sequences from the list were randomly selected to be used for the base DNABERT input. The sequences were truncated to 100'000 base pairs each to reduce computational load. For all ten sequences, gene locations were extracted from the genes list and stored as 3-tuples consisting of the start, end and type of the gene. The start and end values were then passed into an Interval Tree for gene presence which held a 1 along the interval if any gene spanned that range, or a 0 if no gene was present across the interval. Similarly, a second Interval Tree for gene type was created using the same tuples, except instead of storing a single binary value, it stored a set of strings corresponding to each gene type that spanned that interval. This allowed intervals, even single base pair intervals, to hold multiple gene types at a time, allowing for overlapping genes to be captured in this format, exemplifying the multi-label classification nature of this problem.

Next, the raw sequence was "chunked" into 50% overlapping windows of 512 6-mers, making each window 3'072 base pairs long with a step size of 1'536 base pairs. Each window chunk was then fed into the model, the original "zhihan1996/DNA_bert_6" pretrained base model, and its embeddings were collected along with the chunk's corresponding k-mer labels. The labels were created by comparing the starting base pair of each k-mer with the values stored in both Interval Trees at that index and appending the binary value from the presence tree and the set of gene types from the type tree to two separate lists. The labels from these two lists were then paired with the embeddings from the model and appended to one of two embeddings lists, embeddings for gene presence and embeddings for gene type. Each embeddings list consisted of a tuple for each sequence chunk containing its embeddings as one element and its labels as the other.

2.3 Base DNABERT Evaluation

The final embeddings and corresponding labels were passed into two random forest models that were trained on the embeddings to determine if any patterns were identified from the DNABERT output. The models were fed 80% of the data for training and 20% for evaluation. The architecture used for both was 100 estimators and a balanced class weight strategy. Gene detection predictions were evaluated by measuring accuracy and the F1-score as well as generating a classification report and a Confusion Matrix.

Gene type predictions were evaluated by first binarizing the string type labels into multi-hot encoded NumPy arrays and eliminating any gene types that never appeared in the input. Then, the second random forest, identical to the first, was trained and evaluated using Hamming loss, micro- and macro-averaged F1-scores metrics, as well as generating a classification report and confusion matrices for each gene type present.

2.4 Custom Model Training and Evaluation

To prepare data for fine-tuning inputs, the entire preprocessing pipeline was repeated using the same 113 sequences; however, the sequences were not truncated this time to preserve gene types and placement patterns that may exist toward the back end of each sequence. Additionally,

instead of randomly choosing sequences, the sequences were sorted in ascending order by percentage of CDS genes and only the first ten, the ten least CDS-dominant sequences, were used as inputs. A gene detection DNABERT model was created using the same pretrained base from "zhihan1996/DNA_bert_6" with 2 output classes, for 0 or 1, denoting the presence of a gene for each token. A gene classification DNABERT model was also created from the same pretrained base and set to have output classes equal to the number of gene types remaining after binarization and multi-hot encoding of the input data; this means that the number of output classes varies depending on the extant gene types among the bacterial genomes it is trained on; for this experiment, there were eight gene types in the data, and therefore, eight output classes for the model. Furthermore, the classification model had a custom classification header inheriting the main BertForTokenClassification header but overriding the cross-entropy loss function with a binary cross-entropy with logits to support multi-label predictions.

Both models were trained for two epochs due to time and compute limitations and used a training batch size of 12 and an evaluation batch size of 4. The main training metric for gene detection was F1-score while the main training metric for gene classification was macro-averaged F1-score, treating each class as equally important. Additionally, accuracy, recall, and precision were measured for gene identification training and evaluation, with the addition of a confusion matrix for evaluation, as well. Training for gene detection measured micro- and macro-averaged F1-scores, as well as micro- and macro-averaged precision and recall scores, as well. All of these metrics were also used for gene classification evaluation, with the addition of a classification report and confusion matrices per gene type. Both models underwent an 80%/20% split between training and evaluation subsets.

3 Results

The results of each experiment are largely due to the time constraints around the project as well as the computational resources allotted by the researcher. All computation was performed using only a personal laptop with an NVIDIA GeForce RTX2070 with Max-Q GPU; all model predictions were made using the GPU to improve runtime. Regardless of hardware acceleration, the entire experiment took over 24 hours to run on the ten genomic input sequences. Data quality was also a limiting factor as even when selecting the ten least CDS-dominant sequences, CDS genes still occupied over 97% of the total dataset. Table 1 shows the full distribution of gene types used for the fine-tuned model.

Additionally, the gene-to-non-gene k-mer ratio was 10:1, likely introducing a bias toward k-mers that are part of a gene; however, the fine-tuned gene presence identification model seemed to overcome this bias, as shown in section 3.2.1.

Table 1. Gene type distributions for fine-tuned model inputs

Gene Type Distribution		
Gene Type	% in Fine-Tuned Dataset	% in Base Dataset
CDS	97.87%	99.21%
rRNA	1.41%	0.65%
tRNA	0.27%	0.14%
ncRNA	0.03%	0%
tmRNA	0.02%	0%
regulatory	0.33%	0%
Protein bind	0.05%	0%
Mobile element	0.02%	0%

CDS genes make up almost 98% of total input dataset while rRNA, the next most prevalent gene type, only makes up 1.41% of the dataset. The base model received sequences that only had CDS, rRNA, and tRNA genes with a distribution heavily favoring CDS genes.

3.1 Base DNABERT Performance

The Base model was not able to provide embeddings that allowed the random forest model to recognize any significant patterns from the data. For both tasks, the model predicted gene over non-gene and CDS over any other gene type nearly 100% of the time.

On the surface, the model appears to have performed well, achieving an F1-score of 0.94 and an accuracy measure of 0.89 for identification. Upon closer inspection, it is revealed that by predicting "gene" every time, the distribution of non-gene k-mers was too low to significantly lower the performance metric, and therefore, the model never learned to differentiate gene from non-gene k-mers.

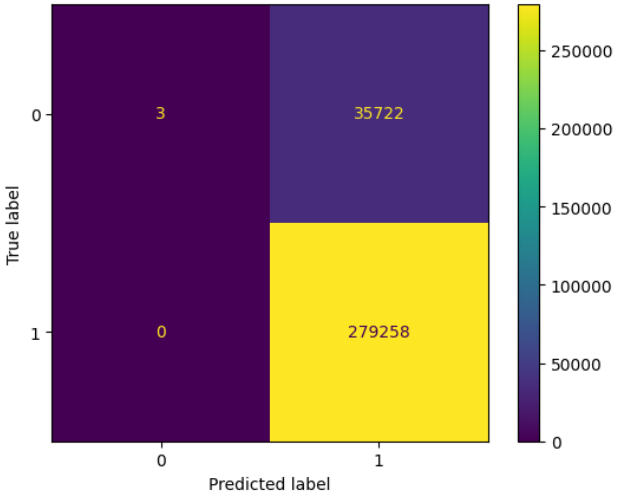

Fig. 1. Confusion matrix for random forest gene identification evaluation of Base DNABERT embeddings. The random forest model nearly always predicted gene (1).

Similarly, the gene classification evaluation seemed promising due to its Hamming loss of 0.04 and micro-averaged F1-score of 0.94; however, the true metric, the macro-averaged F1-scoreof 0.35 revealed that the model was not successfully differentiating classes but was specializing in only identifying one of them. This time, the model predicted CDS for almost every gene, and for the most part, was right almost every time considering CDS genes outnumbered the other two types present 125:1.

Table 2. Random forest classification evaluation metrics

Base DNABERT Classification Report				
Measure	Precision	Recall	F1-score	Support
CDS	**0.88**	**1.00**	**0.94**	**278278**
rRNA	0.69	0.06	0.11	1843
tRNA	0.00	0.00	0.00	381
Micro-average	0.88	0.99	0.94	280502
Macro-average	**0.53**	**0.35**	**0.35**	**280502**
Weighted average	0.88	0.99	0.93	280502
Samples average	0.88	0.88	0.88	280502

The random forest model nearly always predicted CDS and almost never predicted other gene types for each k-mer.

3.2 Fine-Tuned Models Performance

The two fine-tuned models were able to perform significantly better than the base model, partially because of the shift from CDS gene prevalence in the input set, although CDS genes still occupied over 97% of the data. Additionally, despite an increase in disparity between gene-to-non-gene k-mers, from 7:1 to 10:1, the identification model was able to perform accurately without defaulting to the more dominant prediction.

3.2.1 Gene Identification

The trained gene identification model received an accuracy measure of 0.94, precision of 0.97, recall of 0.97, and F1-score of 0.97. The confusion matrix in Figure 2 makes it easy to see that it excelled at identifying gene tokens but still managed to perform moderately well at identifying non-gene tokens.

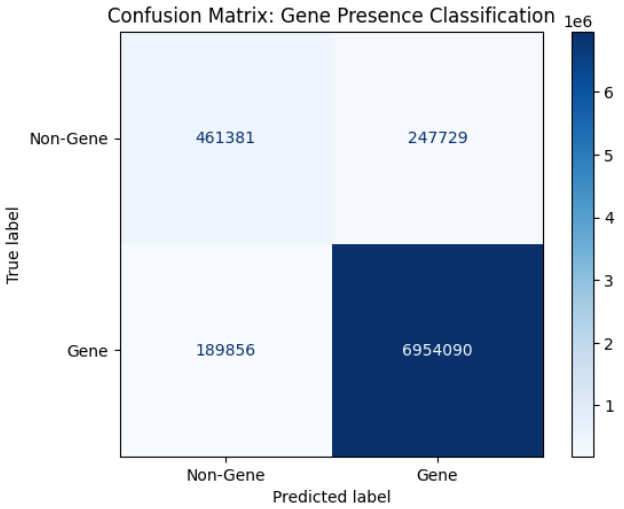

Fig. 2. Confusion matrix for fine-tuned gene identification DNABERT model. Despite a ratio of 10:1 gene-to-non-gene k-mers, the model predicted accurately at a rate of 11:1.

3.2.2 Gene Classification

The trained gene classification model excelled at identifying CDS genes, as well as the second most common type, rRNA, receiving F1-scores of 0.97 and 0.92, respectively. The performance for tRNA genes dropped dramatically to an F1-score of 0.32, and the five remaining gene classes were never predicted, thus all receiving scores of 0 for all metrics. Despite these metrics hindering the macro-averaged F1-score to 0.28, it is obvious that the model was able to identify patterns for the more prevalent gene types and was especially effective at identifying rRNA genes even though they only made up 1.41% of the dataset.

Table 3 gives the full range of metrics for the evaluation and Table 4 lays out each of the eight confusion matrices in table format. The confusion matrices makes it clear that the model did at least attempt to learn patterns around ncRNA, tmRNA, regulatory genes, protein-binding genes, and mobile elements, but due to the low prevalence of these types within the data, there were none to verify in the evaluation set after the 80%/20% split.

Table 3. Fine-Tuned Gene Classification - Classification Report

Multi-label Token Classification Report				
	Precision	Recall	F1-score	Support
CDS	**0.96**	**0.97**	**0.97**	**7058753**
rRNA	**0.91**	**0.93**	**0.92**	**95576**
tRNA	**0.51**	**0.23**	**0.32**	**20962**
ncRNA	0.00	0.00	0.00	2919
tmRNA	0.00	0.00	0.00	1406
regulatory	0.00	0.00	0.00	25270
Protein bind	0.00	0.00	0.00	3275
Mobile element	0.00	0.00	0.00	1838
Micro-average	0.96	0.96	0.96	7209999
Macro-average	**0.30**	**0.27**	**0.28**	**7209999**
Weighted average	0.96	0.96	0.96	7209999
Samples average	0.89	0.88	0.88	7209999

The model clearly identified patterns among the three more represented classes of genes, despite the main metric being insufficient.

Table 4. Fine-Tuned Gene Classification - Confusion Matrices

Multi-label Token Classification Confusion Results				
Gene Type	**True Negative**	**False Negative**	**False Positive**	**True Positive**
CDS	**532768**	**261535**	**198828**	**6859925**
rRNA	**7749070**	**8410**	**6231**	**89345**
tRNA	**7827389**	**4705**	**16088**	**4874**
ncRNA	7850137	0	2919	0
tmRNA	7851650	0	1406	0
regulatory	7827786	0	25270	0
Protein bind	7849781	0	3275	0
Mobile element	7851218	0	1838	0

The model performed well at predicting CDS, rRNA, and tRNA genes, but still attempted to predict the other five rarer types despite them not appearing in the evaluation set.

Conclusion

The results show that even extremely limited fine-tuning with just ten reference genomes is enough to retrain DNABERT to accurately capture complex patterns and feature interactions between genome sequences, millions of base pairs long from a domain of life entirely separate from that which it was trained on. Given minimal computational power, just one GPU on a personal gaming laptop, both models were able to achieve micro-averaged F1-scores for both detection and classification at or over 0.96. When averaging only the F1-scores for just the three gene types that were predicted by the classification model, CDS, rRNA, and tRNA, a new macro-averaged F1-score can be evaluated at 0.74, which is not ideal, but still shows that the model was able to identify gene patterns

with highly limited input. Moreover, the mere fact that the fine-tuned models did not simply default to the most prevalent label type as the base model did shows that some information was learned and is present in the parameters of the model.

It is also possible that the choice to ignore unconfirmed gene types, such as miscellaneous types or generic "gene" labels may have skewed the distribution in favor of CDS more so than if all the gene labels were used as they appeared in the original data files. Running a new experiment without the removal of these genes could prove to be fruitful and may even show improvement on the part of DNABERT to differentiate between a larger number of classes.

Regardless of this project's limitations, there is enough evidence to suggest that DNABERT can be used reliably to resolve the gap in representation for genomic analysis on bacterial sequences via machine learning. DNABERT's ability to identify key feature interactions is invaluable for genomic analysis due to the heightened complexity of the problem and the importance of understanding some of the most pervasive genomes known to human sciences.

Discussion

As previously stated, the biggest hindrances to performance were data quality and computational resources. The low-hanging fruit to improve this project is to simply curate the inputs to have more sequences trained, and more importantly, to have better distributions for the rarer sequences present. As noted, even having as low as 1.4% presence in the training set allowed the model to accurately and reliably identify those rRNA genes. These findings clearly outline the power of transformer models, and their ability to adapt to input types far different from those they were initially trained on.

With multiple GPUs on a server, likely through a professional network or a cloud service, the runtime limitations imposed on this project could be completely resolved at its current scale, or in the same amount of time, the model could be trained on thousands of sequences instead of ten. This is beneficial not only for increasing the number of patterns to learn from, but also to give greater representation to the genes that were less prevalent in the dataset, resolving class imbalance issues seen in this study. One must consider that even if CDS genes still maintain over 95% of the dataset, with enough genes, even 1% of the dataset could be enough to allow DNABERT to learn significant patterns because bacterial genes are only so long on average, generally from 800 to 1600 base pairs if based on the 113 sequences used in this experiment. Also, if more time is allotted for a project of this caliber, training could also be run for more than two epochs in order to refine the models' parameters on the micro-averaged F1-score metric to lower loss further.

Furthermore, this project sets the groundwork for thorough downstream analyses with transformers or even other machine learning models by efficiently extracting spurious amounts of features from the GBFF input files. Another worthwhile extension to this experiment is to pass in some of these features, such as gene products and strand information to potentially improve DNABERT's ability to recognize deeper feature interactions. There is a lot more to be gained from exploring bacterial genomes with powerful tools like transformers, and DNABERT and its successors are only going to improve.

Code Availability

All code used for this experiment is available at: https://github.com/JustinMorera/Bacterial-Gene-Identification-ML-Model

References

Ji, Y., Zhou, Z., Liu, H., & Davuluri, R. V. (2021). DNABERT: Pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Bioinformatics, 37(15), 2112–2120. https://doi.org/10.1093/bioinformatics/btab083

NCBI Resource Coordinators. Database resources of the National Center for Biotechnology Information. Nucleic Acids Research. 2016 Jan 4;44(D1):D7–D19. doi:10.1093/nar/gkv1290

O'Leary NA, Wright MW, Brister JR, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Research. 2016 Jan 4;44(D1):D733–D745. doi:10.1093/nar/gkv1189

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Yanrong Ji, Zhihan Zhou, Han Liu, Ramana V Davuluri, DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome, Bioinformatics, Volume 37, Issue 15, August 2021, Pages 2112–2120, https://doi.org/10.1093/bioinformatics/btab083

Zhou, Z., Ji, Y., Li, W., Dutta, P., Davuluri, R., & Liu, H. (2023). DNABERT-2: Efficient foundation model and benchmark for multi-species genome [Preprint]. arXiv. https://arxiv.org/abs/2306.15006

James, T., Williamson, B., Tino, P., & Wheeler, N. (2025). Whole-Genome Phenotype Prediction with Machine Learning: Open Problems in Bacterial Genomics. arXiv preprint arXiv:2502.07749.