

deepFEPS: Deep Learning-Oriented Feature Extraction for Biological Sequences

Hamid Ismail (HI), Ph.D. (Corresponding author)

Department of Computational Data Science and Engineering

North Carolina A&T State University

East Market Street, Greensboro, North Carolina 27411, US.

Email: hismail@ncat.edu

Marwan Bikdash (MB), Ph.D.

Department of Computational Data Science and Engineering

North Carolina A&T State University

East Market Street, Greensboro, North Carolina 27411, US.

Abstract

Background:

Machine- and deep-learning approaches for biological sequences depend critically on transforming raw DNA, RNA, and protein FASTA files into informative numerical representations. However, this process is often fragmented across multiple libraries and preprocessing steps, which creates a barrier for researchers without extensive computational expertise. To address this gap, we developed deepFEPS, an open-source toolkit that unifies state-of-the-art feature extraction methods for sequence data within a single, reproducible workflow.

Results:

deepFEPS integrates five families of modern feature extractors —k-mer embeddings (Word2Vec, FastText), document-level embeddings (Doc2Vec), transformer-based encoders (DNABERT, ProtBERT, and ESM2), autoencoder-derived latent features, and graph-based embeddings—into one consistent platform. The system accepts FASTA input via a web interface or command-line tool, exposes key model parameters, and outputs analysis-ready feature matrices (CSV). Each run is accompanied by an automatic quality-control report including sequence counts, dimensionality, sparsity, variance distributions, class balance, and diagnostic visualizations. By consolidating advanced sequence embeddings into one environment, deepFEPS reduces preprocessing overhead, improves reproducibility, and shortens the path from raw sequences to downstream machine- and deep-learning applications.

Conclusions:

deepFEPS lowers the practical barrier to modern representation learning for bioinformatics, enabling both novice and expert users to generate advanced embeddings for classification, clustering, and predictive modeling. Its unified framework supports exploratory analyses, high-

throughput studies, and integration into institutional workflows, while remaining extensible to emerging models and methods. The webserver is accessible at <https://hdismail.com/deepfeeps2/>.

Keywords

Feature extraction, biological sequences, deep learning, sequence embeddings, bioinformatics toolkit, transformer models, Word2Vec, autoencoder, graph embeddings.

Background

Extracting numerical features from biological sequences has long been a challenge, largely because these sequences are inherently textual and must be transformed into meaningful numerical representations before they can be used in machine learning. This process often requires considerable programming and computational expertise. The application of machine learning to biological sequences dates back to 1962, when Margaret Oakley Dayhoff and Richard Eck reconstructed evolutionary relationships using a maximum parsimony method, an approach that can be regarded as an early form of computational inference from molecular data [1]. Neural networks entered the field in 1988, when they were first applied to predict the secondary structure of globular proteins [2]. By the 1990s, hidden Markov models (HMMs) had gained popularity for tasks such as gene prediction, protein family classification, and motif discovery [3].

The scope of machine learning in biological sequence analysis expanded dramatically after the completion of the Human Genome Project in 2003 and the subsequent explosion of sequencing data. Algorithms such as Support Vector Machines (SVMs), Random Forests (RF), and kernel methods were increasingly applied to a wide range of problems, including promoter prediction, splice site recognition, noncoding RNA classification, and protein-protein interaction prediction [4, 5]. These traditional machine learning methods typically relied on features derived from classical sequence descriptors, such as nucleotide or amino acid composition, physicochemical properties, and autocorrelation descriptors [6, 7]. Such handcrafted features provided the foundation for training models that could capture sequence patterns and address a variety of supervised learning problems in bioinformatics.

Building on the foundations of traditional machine learning and feature extraction, the advent of deep learning marked a transformative era in biological sequence analysis beginning around 2015. One of the first breakthroughs was DeepBind, which employed convolutional neural networks (CNNs) to predict protein-DNA and protein-RNA binding, showing that deep architectures could surpass classical motif-based approaches [8]. In the same year, DeepSEA extended CNNs to model the chromatin effects of noncoding variants, laying the groundwork for interpreting genome-wide regulatory activity directly from raw sequence [9]. A few years later, AlphaFold (2018–2021) demonstrated the unprecedented capacity of deep neural networks in structural biology, reaching near-experimental accuracy in protein structure prediction and fundamentally reshaping the field [10]. More recently, advances from natural language processing and representation learning have been extensively adapted to genomics and proteomics. Embedding methods such as Word2Vec and FastText have been applied to biological sequences, treating k-mers as tokens to generate dense distributed representations that capture contextual similarity [11, 12]. Extending beyond word embeddings, Doc2Vec has been utilized to obtain sequence-level representations, allowing entire genes or proteins to be embedded into a continuous space for downstream classification tasks [13]. Unsupervised neural architectures like autoencoders have also been employed to derive latent features from DNA, RNA, and protein sequences, compressing high-dimensional sequence space into lower-dimensional informative representations useful for phenotype prediction and

variant prioritization [14]. In parallel, graph-based embeddings such as Graph2Vec and node2vec have been applied to biological networks and sequence-derived graphs, enabling representation of motifs, domains, and higher-order relationships not easily captured by linear models [15, 16]. The most impactful advances, however, have come from transformer-based models, which leverage self-attention to capture long-range dependencies in sequences. DNABERT introduced k-mer-based pretraining to model DNA as a language, demonstrating robust performance across a wide spectrum of genomic tasks [17]. In proteomics, ProtBERT, part of the ProtTrans project, applied the BERT architecture at scale to billions of protein sequences, yielding embeddings that generalize across diverse prediction tasks [18]. Similarly, ESM-2, a large-scale protein language model, has provided state-of-the-art embeddings and atomic-level predictions of protein structures across diverse protein families [19]. Together, these innovations signify the deep learning revolution in bioinformatics, where embedding-based representation learning and large-scale neural architectures now lie at the core of sequence-based discovery.

With these advances, it has become increasingly clear that bioinformatics needs tools capable of bridging the gap between classical feature engineering and modern representation learning. As a successor to the original FEPS toolkit [20, 7], which has supported users since 2016 with traditional feature extraction methods, we now introduce deepFEPS, a comprehensive platform that consolidates five families of state-of-the-art embedding-based feature extractors for DNA, RNA, and protein sequences. Unlike conventional approaches, deepFEPS integrates language-model-inspired techniques to transform raw biological sequences into rich numerical representations suitable not only for deep learning but also for traditional machine learning frameworks. Its modules encompass Word2Vec, FastText, and Doc2Vec embeddings, transformer-based encoders, autoencoder-derived latent features, and graph-based sequence embeddings. By moving beyond simple k-mer counts and handcrafted descriptors, deepFEPS captures higher-order dependencies and complex sequence relationships, enabling more powerful and generalizable analysis across a wide range of sequence-based bioinformatics applications.

Implementation

deepFEPS is implemented as a modular toolkit that consolidates diverse sequence representation methods within a single, reproducible framework. The system is designed to handle DNA, RNA, and protein FASTA inputs through both a web server and a command-line interface, ensuring accessibility for users with different computational backgrounds. At its core, deepFEPS organizes five families of feature extractors, ranging from classical embedding models to modern transformer architectures, into configurable modules that follow consistent input/output conventions. Each module exposes its key parameters, generates analysis-ready feature matrices in CSV format, and produces automated quality-control reports with visual diagnostics. This unified design allows users to seamlessly compare alternative feature extraction strategies while maintaining reproducibility and interoperability with downstream machine- and deep-learning workflows

Feature extractors

1. k-mer Word2Vec/FastText (sequence token embeddings)

In this approach, biological sequences are first segmented into overlapping k-mers, which serve as tokens. A word embedding model, such as skip-gram or CBOW in Word2Vec [21] or the sub-word-aware FastText [22], is then either trained from scratch or initialized from pretrained weights. The resulting token embeddings are aggregated using pooling strategies such as mean, mean+max, or

CLS embeddings (from transformer-based models), yielding a fixed-length vector representation for each sequence. These embeddings preserve the local compositional context of sequences and provide robust baselines for many downstream tasks. Users can tune key parameters such as k-mer size (k), context window, embedding dimensionality, training epochs, negative sampling strategy, and the choice between pretraining or on-the-fly training to optimize performance.

Word and sub-word embeddings have already demonstrated success in several areas of bioinformatics. For example, k-mer-based Word2Vec models have been used to classify protein families and predict subcellular localization with high accuracy [11]. In genomics, embeddings have enabled promoter and enhancer prediction, as well as the identification of splice sites and other regulatory elements [23]. These successes highlight how embedding-based representations can serve as powerful and versatile building blocks for biological sequence modeling.

2. Doc2Vec (document embeddings for sequences)

Doc2Vec [24] extends the idea of word embeddings by treating each biological sequence as a “document” and learning a sequence-level embedding jointly with its constituent k-mer tokens. Unlike simple k-mer averaging, this method captures both global context and local composition, making it particularly effective for identifying long-range motifs or integrating compositional and positional information within a sequence. Performance depends on parameters such as the k-mer size (k), context window, embedding dimensionality, number of training epochs, and the choice between Distributed Memory (DM) or Distributed Bag of Words (DBOW) training modes.

Doc2Vec has shown strong potential in bioinformatics applications where understanding the entire sequence context is crucial. For instance, it has been used to predict protein functions and classify protein families, leveraging global representations to distinguish subtle functional differences [13]. In genomics, Doc2Vec embeddings have been applied to enhancer and promoter prediction and to capture regulatory sequence activity [25]. These applications demonstrate that Doc2Vec provides a powerful complement to word-level embeddings, especially in tasks that benefit from holistic sequence representation.

3. Autoencoder-derived features

In this approach, biological sequences are first converted into numerical forms, such as k-mer bag-of-words profiles or one-hot encoded windows and then passed through a feed-forward autoencoder. Instead of keeping all the raw input dimensions, the model learns to compress the information into a low-dimensional bottleneck layer, and these activations become the exported features. This compression not only reduces redundancy but also acts as a natural denoiser, emphasizing the most informative aspects of the data. Users can fine-tune the model by adjusting the latent dimensionality, layer widths, activation functions, and regularization strategies such as dropout or normalization, along with training settings like epochs and batch size. Because the mapping is learned directly from the provided dataset (or from a specific corpus) the resulting features are context-aware and tailored to the biological problem at hand.

Autoencoders have already shown promise across many areas of bioinformatics. For instance, they have been used to denoise single-cell RNA-seq data, improving downstream clustering and differential expression analysis [26]. In proteomics, autoencoder-derived features have helped in protein function prediction and subcellular localization [27]. They have also been applied in variant effect prediction, learning compressed sequence features that generalize across species and

contexts [28]. These successes highlight the flexibility of autoencoders in uncovering complex biological patterns that traditional handcrafted features may overlook.

4. Graph-based embeddings

Graph-based embeddings approach sequence modeling by first representing biological data as a graph; for example, building edges from k-mer co-occurrence within sequences or from sequence-similarity relationships across datasets. Once the graph is constructed, node or graph embeddings can be learned using algorithms such as node2vec-style random walks, which capture relational structure that complements the linear token order. These node embeddings are then aggregated across a sequence (using operations like mean pooling or attention-based weighting) to produce a fixed-length descriptor suitable for downstream tasks. Key parameters include the definition of edges or windows, walk length, number of random walks, and the embedding dimensionality.

Graph embeddings have found increasing success in bioinformatics, particularly where relational dependencies are as important as compositional ones. For example, k-mer co-occurrence graphs combined with node2vec embeddings have been used for viral genome classification and host-pathogen interaction prediction [29, 30]. In proteomics, graph-based methods have proven powerful for protein-protein interaction (PPI) prediction and functional annotation, since they naturally capture network relationships among biomolecules [31]. These examples illustrate that graph embeddings enrich sequence representation by uncovering structural and contextual information often missed by linear models.

5. Transformer-based encoders (pretrained protein and nucleotide models)

Transformer-based encoders represent the newest generation of sequence embedding methods, drawing on architectures originally developed for natural language processing. For proteins, models such as ESM-2 provide powerful pretrained embeddings, while for DNA and RNA, k-merized transformer models like DNABERT and its variants have become widely used. In practice, sequences are tokenized using the model’s native scheme, with built-in safeguards for long inputs, such as safe truncation or chunking on the server side, to ensure scalability. The resulting embeddings can be aggregated with configurable pooling strategies (CLS, mean, or mean+max) to produce a per-sequence vector representation. By leveraging transfer learning from massive unsupervised sequence corpora, transformer encoders often deliver the strongest out-of-the-box performance across a wide range of functional prediction tasks.

Transformers have already achieved notable success in bioinformatics. ESM models have been applied to protein structure prediction, mutation effect analysis, and function annotation, often rivaling experimental pipelines in speed and accuracy [19, 32]. In genomics, DNABERT has been used for tasks such as promoter and enhancer prediction, splice site recognition, and chromatin state classification, where its ability to model long-range dependencies in DNA provides a major advantage [17]. Recent studies have further demonstrated the versatility of transformer-based language models in proteomics, particularly for post-translational modification (PTM) prediction. For instance, LMPhosSite applied embeddings from local sequence windows alongside a pretrained protein language model to accurately predict phosphorylation sites [33]. Building on this, researchers have shown that integrating embeddings from multiple protein language models can significantly improve the prediction of O-GlcNAc modification sites, highlighting the benefit of ensemble representation learning [34]. Similarly, LMCrot leveraged interpretable transformer-based embeddings to enhance crotonylation site prediction [35], while CaLMPhosKAN introduced a hybrid approach combining codon-aware and amino-acid-aware embeddings with wavelet-based

Kolmogorov-Arnold networks, achieving state-of-the-art performance for phosphorylation site prediction [36]. Together, these works illustrate how transformer-based protein models are not only powerful for structural biology and general function prediction but are also enabling specialized, high-accuracy predictors for PTMs and regulatory biology.

Results

Input and preprocessing

deepFEPS accepts multi-FASTA inputs for DNA, RNA, or protein sequences. To prepare data for supervised modeling, the FASTA definition line can be formatted as:

```
>ID|label
```

This ensures that the sequence ID is recorded in the first column of the output CSV file, while the label appears in the last column, making the dataset immediately suitable for supervised modeling. Additional options are available depending on the chosen feature extractor.

The web server is designed for small to moderate datasets, making it ideal for quick tests or exploratory analyses. For larger datasets and more advanced workflows, we recommend downloading deepFEPS and running it via the command-line interface (CLI), which supports batch processing and more flexible configurations. Detailed hardware and software requirements for local installation are provided in the GitHub repository.

Output and automatic QC

Each run produces an analysis-ready CSV in which rows correspond to sequences and columns to features, with optional identifier and label fields when supplied. To make the output immediately interpretable, the web interface provides a quality-control dashboard, as shown in **Figure 2**, that reports the number of sequences processed, the feature dimensionality, matrix size on disk, missing-value counts (imputed as zeros), and overall sparsity; key descriptors for checking whether the dataset is of the expected scale. A lightweight preview table displays the first few rows and capped columns of the feature matrix, allowing users to quickly confirm that identifiers are in the right place, numeric ranges look reasonable, constant columns are absent, and no label information has leaked into the feature set.

Beyond the preview, diagnostic visualizations provide deeper insights into data quality. The row-norm distribution (**Figure 3**) shows the L2 magnitude of each sequence’s feature vector. A tight cluster around a central peak suggests consistent scaling across samples, whereas a long right tail signals that some sequences dominate with unusually large values, and a heavy left tail or isolated low bars can reveal near-empty vectors from filtering or tokenization mismatches. Corrective steps include per-row normalization or clipping. The feature-variance distribution (**Figure 4**) summarizes variability across sequences for each feature. Features with near-zero variance contribute little to discrimination and can be removed, while those with very high variance may highlight biologically informative heterogeneity or, in some cases, artifacts of extractor settings. Users can decide whether to apply variance filtering or dimensionality reduction methods such as PCA based on this view.

When labels are present, the class distribution pie chart (**Figure 5**) and table provide a quick check on balance. Equal slice sizes suggest balanced data, while skewed proportions reveal class

imbalance that may bias models toward the majority class. In such cases, strategies like class weighting, resampling, or stratified evaluation are recommended. Finally, the heatmap preview (**Figure 6**) offers a visual snapshot of a small slice of the feature matrix after normalization. Vertical bands of uniform color indicate low-variance or constant features, horizontal bands suggest duplicate or highly similar sequences, and block-like patterns reveal correlated groups of features or motif signals. Users can use these patterns to decide whether additional preprocessing, such as feature pruning, deduplication, or correlation filtering, is needed.

Together, these summaries and visualizations do more than describe the dataset: they actively guide users in diagnosing scaling issues, identifying redundant or noisy features, checking label distributions, and spotting structural patterns. By interpreting these cues, users can make principled choices about normalization, feature selection, class balancing, and dimensionality reduction, ensuring more reliable and stable downstream modeling.

Web Server and Command-Line availability

deepFEPS is available as both an interactive web server (<https://hdismail.com/deepfeeps2/>) (see **Figure 1**) and a command-line interface (CLI) for scripted, large-scale, or high-throughput workflows. The web server is designed for exploratory use and parameter tuning on small to medium datasets, whereas the CLI exposes advanced options, supports CPU/GPU execution where available, and integrates smoothly with institutional pipelines and workflow managers. Source code and the CLI are hosted at <https://github.com/hamiddi/deepFEPS>. The web service accepts compressed or plain FASTA uploads, returns a download link for the resulting CSV feature matrix, and automatically records a parameter manifest to facilitate reproducibility.

Performance considerations

Computational requirements scale with the choice of extractor and the size of the corpus. Transformer encoders typically incur the highest runtime and memory demand because of large model weights and tokenization overhead, but they often deliver the strongest transfer-learning baselines. Autoencoders and k-mer token embeddings are substantially lighter and easier to scale horizontally across CPUs or modest GPUs. For graph-based embeddings, cost is dominated by graph density and random-walk settings (e.g., window length and number of walks), which trade runtime against representation fidelity. For very large or repeated submissions (and when fine control over hardware, catching pretrained models, batching, or mixed precision is desired) the command-line interface is recommended.

Limitations and future directions

Despite its breadth, deepFEPS has several limitations. Transformer encoders have fixed positional capacities; processing very long sequences often requires chunking or sliding windows, which can attenuate long-range dependencies. We plan to expose explicit windowing, stride, and aggregation policies (e.g., mean, attention-weighted, or learned pooling) and to report effective context coverage. Graph-based embeddings remain sensitive to graph construction heuristics—including k-mer co-occurrence definitions, edge weighting, and thresholding—and their performance can vary with graph density; incorporating domain-specific priors (e.g., known interaction networks or pathway constraints) is a priority. Finally, the current QC dashboard provides lightweight, practical indicators; forthcoming modules will add formal diagnostics such as mutual information

with labels, redundancy and collinearity measures, stability across resamples, and drift detection, together with exportable reports. Additional roadmap items include enhanced memory/performance profiling, deterministic execution for strict reproducibility, and broader support for pretrained models and offline caching.

Discussion

deepFEPS builds on and extends earlier sequence feature extraction frameworks. Classical tools such as iFeature [6] and dna2vec [23] provided access to predefined descriptors or word embedding-based representations, but were limited in scope and flexibility. Our earlier toolkit, FEPS [7, 20], offered a webserver and software package for extracting handcrafted protein sequence descriptors, supporting researchers with traditional machine-learning workflows for nearly a decade. While FEPS lowered barriers to protein sequence analysis, it was designed around classical feature engineering approaches.

In contrast, deepFEPS can be viewed as the next-generation successor to FEPS, consolidating five families of modern embedding methods—including k-mer embeddings, document-level models, autoencoders, graph embeddings, and transformer-based encoders—into a single, reproducible workflow. By bridging traditional descriptors with state-of-the-art representation learning, deepFEPS unifies both paradigms in one environment and removes the need to switch between multiple libraries or fragmented pipelines.

The platform is broadly applicable across sequence analysis tasks. By lowering the barrier to advanced embeddings, deepFEPS enables promoter and enhancer prediction, protein function classification, noncoding RNA characterization, and post-translational modification site prediction. It also supports large-scale applications such as genome annotation and protein family clustering, where reproducible, high-quality embeddings are essential for downstream machine- and deep-learning models.

Although we did not include a full benchmarking study here, preliminary applications on protein family datasets showed that transformer-based embeddings captured richer discriminative features compared to k-mer baselines, consistent with recent results from large protein language models [19]. These findings suggest that deepFEPS not only streamlines access to advanced embeddings but also empowers researchers to evaluate, within a single platform, which feature extraction strategy best aligns with their biological question.

Conclusions

deepFEPS brings modern sequence representation learning into a single, coherent tool that is equally effective as an interactive web service and as a scriptable command-line system. By unifying diverse extractors under consistent input/output conventions, producing reproducible, timestamped outputs, and providing immediate quality-control diagnostics, it shortens the path from raw DNA, RNA, and protein sequences to analysis-ready feature matrices. As the next-generation successor to the FEPS toolkit, deepFEPS lowers the practical barrier to advanced embeddings and supports routine bioinformatics, high-throughput screening, and exploratory computational biology. Its extensible design ensures continued relevance as new representation learning methods emerge, making it a valuable resource for both experimental and computational

researchers. The platform is freely available as a web server at <https://hdismail.com/deepfeeps2/> and as an open-source command-line package at <https://github.com/hamiddi/deepFEPS>.

Availability and requirements

- **Project name:** deepFEPS
- **Project home page:** <https://hdismail.com/deepfeeps2/> (web server)
- **Source code repository:** <https://github.com/hamiddi/deepFEPS>
- **Operating system(s):** Platform independent (tested on Linux, macOS, and Windows)
- **Programming language:** Python (≥ 3.8)
- **Other requirements:**
 - Dependencies managed via requirements.txt or environment.yml (e.g., PyTorch ≥ 1.11 , NumPy, scikit-learn, gensim, transformers, networkx, matplotlib)
 - For transformer-based models, GPU acceleration (CUDA-enabled device) is recommended but not required
- **License:** GNU General Public License v3.0 (GPL-3.0)
- **Any restrictions to use by non-academics:** None (open-source and freely available to all users)

List of abbreviations

- BERT: Bidirectional Encoder Representations from Transformers
- CLI: Command-Line Interface
- CNN: Convolutional Neural Network
- CSV: Comma-Separated Values
- DBOW: Distributed Bag of Words (Doc2Vec training mode)
- DM: Distributed Memory (Doc2Vec training mode)
- DNABERT: DNA Bidirectional Encoder Representations from Transformers
- ESM: Evolutionary Scale Modeling (protein transformer family)
- FEPS: Feature Extraction from Protein Sequences
- GPU: Graphics Processing Unit
- HMM: Hidden Markov Model
- ML: Machine Learning
- NLP: Natural Language Processing
- PCA: Principal Component Analysis
- PPI: Protein–Protein Interaction
- PTM: Post-Translational Modification
- QC: Quality Control
- RF: Random Forest
- ROC: Receiver Operating Characteristic
- SVM: Support Vector Machine
- Word2Vec: Word-to-Vector model (distributed embeddings)

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The web server deepFEPS2 is publicly available at <https://hdismail.com/deepfeeps2/>.

The source code and additional resources are available at <https://github.com/hamiddi/deepFEPS>.

Competing interests

The authors declare that they have no competing interests.

Funding

No funding was received for this work.

Authors' contributions

HI conceived and designed the study, implemented the computational pipeline, and drafted the manuscript. MB contributed to the conceptual framework, provided critical revisions, and supervised the research process. Both authors read and approved the final version of the manuscript.

Acknowledgements

The authors thank colleagues and collaborators at North Carolina A&T State University and BioAGTC for their support and valuable feedback during the preparation of this work.

Authors' information

Hamid Ismail (HI), Ph.D., is the Administrator of the Research High-Performance Computing Center and a Lecturer at North Carolina A&T State University (ORCID: 0000-0002-2690-5655)

Marwan Bikdash (MB), Ph.D., is a Professor and Director of the Department of Computational Data Science and Engineering at North Carolina A&T State University.

deepFEPS: Deep Learning-Oriented Feature Extraction for Biological Sequences

DeepFEPS is a high-performance bioinformatics platform for extracting advanced sequence-based features from DNA, RNA, and protein data. It integrates modern machine learning and deep learning techniques to transform raw biological sequences into rich numerical representations suitable for classification, clustering, and predictive modeling.

Each feature extractor below offers an advanced way of representing biological sequences — from sequence embedding models such as Word2Vec, FastText, and Doc2Vec, to Transformer-based architectures, Autoencoder-derived features, and Graph-based embeddings. These deep learning and graph representation techniques can capture complex sequence patterns and relationships beyond simple k-mer counts, enabling more powerful analysis for functional annotation, motif discovery, and predictive modeling.

Simply select the method that best fits your research goals, upload your sequences, configure the parameters, and download your processed features.

Want to try with sample sequences? Download example FASTA files (DNA & Protein) as a zip and upload to any extractor. [Download examples \(.zip\)](#)

Need advanced features or support for very large datasets?
For heavy workloads and full argument control, please use the **deepFEPS command-line version** from the official GitHub repository. [Open GitHub repo](#)

Autoencoder features

Learned compressed representations using autoencoders on k-mer BoW or fixed one-hot encodings.

[Open](#)

Doc2Vec embeddings

Sequence-as-document embeddings (PV-DM / PV-DBOW) over k-mers; strong global context vectors.

[Open](#)

Graph embeddings

k-mer graph embeddings (DeepWalk/Node2Vec/Graph2Vec) pooled to fixed-size features.

[Open](#)

Transformer embeddings

ProtBERT / ESM2 / DNABERT with configurable pooling and device.

[Open](#)

Word2Vec / FastText

Word2Vec or FastText on k-mers with flexible pooling strategies.

[Open](#)

Figure 1. Screenshot of the deepFEPS web server interface displaying the five groups of feature extractors. Each group corresponds to a family of methods (k-mer embeddings, document-level embeddings, autoencoder-derived features, graph-based embeddings, and transformer-based models) allowing users to select and configure pipelines tailored to DNA, RNA, or protein sequences.

Status: done

[Download features.csv](#)

Log

Feature summary

Sequences: 200
 Numeric features: 128
 File size: 541752 bytes
 Missing values treated as 0: 0
 Sparsity (zeros / numeric cells): 0.000

Preview (first 8 rows × up to 10 columns)

ID	EMB_1	EMB_2	EMB_3
649121578	-0.0029015224426984787	0.023230578750371933	-0.0188292972743
426348349	-0.0028853584080934525	0.02319679781794548	-0.0188798159360
40806160	-0.0029602516442537308	0.023578006774187088	-0.0200736448168
60834865	-0.0029245521873235703	0.023587696254253387	-0.0200677588582
355754119	-0.0019949879497289658	0.02295968122780323	-0.0200148690491
6018474	-0.0027818065136671066	0.0232611745595932	-0.0186398066580
55741799	-0.002956647425889969	0.023628953844308853	-0.0200178213417
825639	-0.002820288762450218	0.023593619465827942	-0.0198700614273

Figure 2. A summary panel reports the number of sequences, features per sequence, file size, missing-value counts, and sparsity, while a lightweight table displays the first rows and columns

of the feature matrix. Together, these provide a quick check that identifiers, scaling, and basic structure match expectations before downstream modeling.

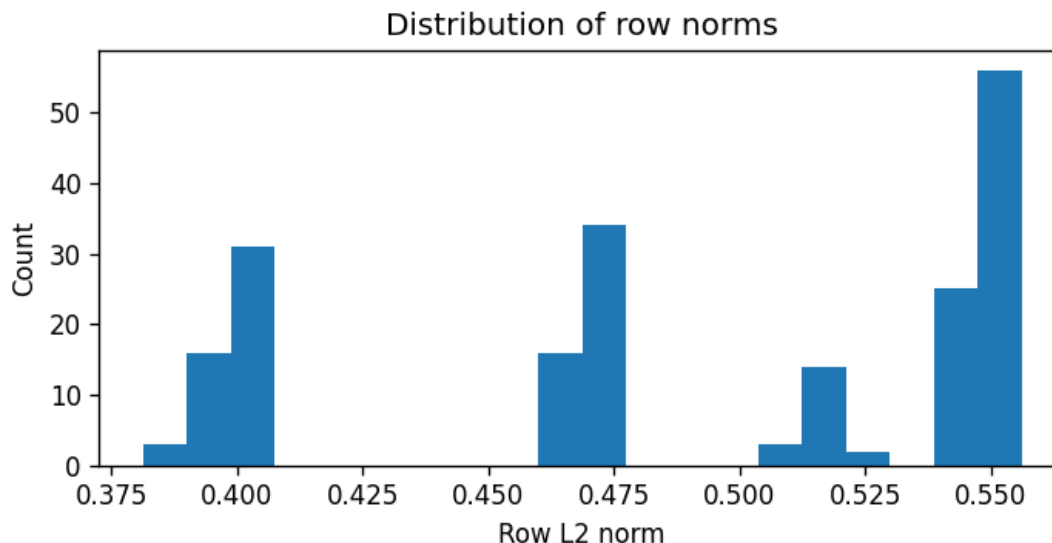


Figure 3. Histogram of per-sequence L2 norms showing the overall magnitude of feature vectors. A sharp central peak indicates consistent scaling across samples, while heavy tails or multimodality suggest outliers, heterogeneous inputs, or extractor issues. Small norms may flag near-empty representations; large norms may indicate scaling drift.

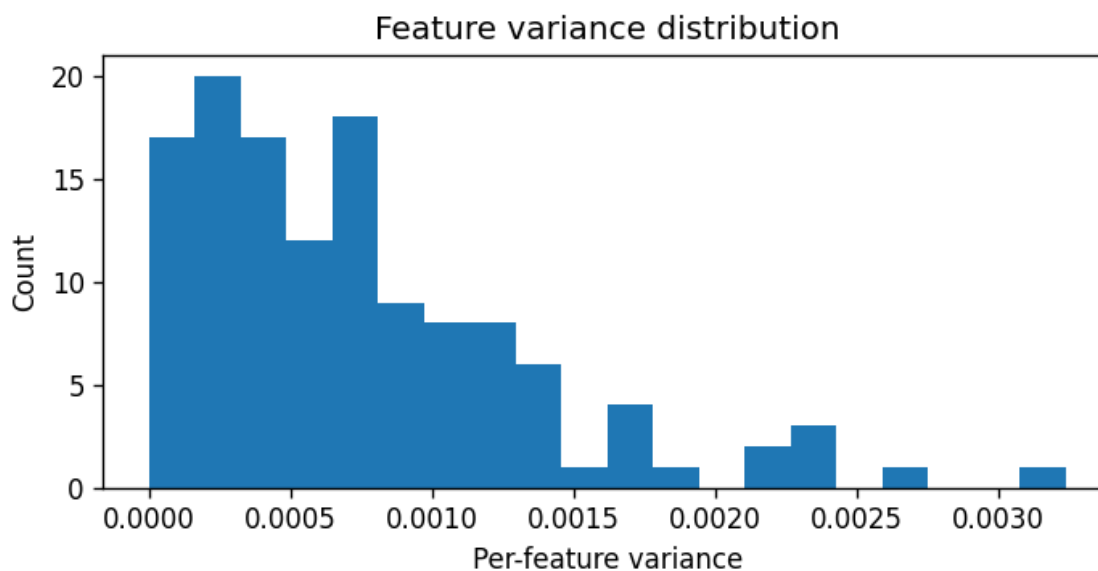


Figure 4. Histogram of feature variances across sequences, used to identify low-information or noisy features. Near-zero variance features contribute little and can be removed, while high-variance features may capture biologically relevant heterogeneity or artifacts. The distribution helps guide dimensionality reduction or variance filtering.

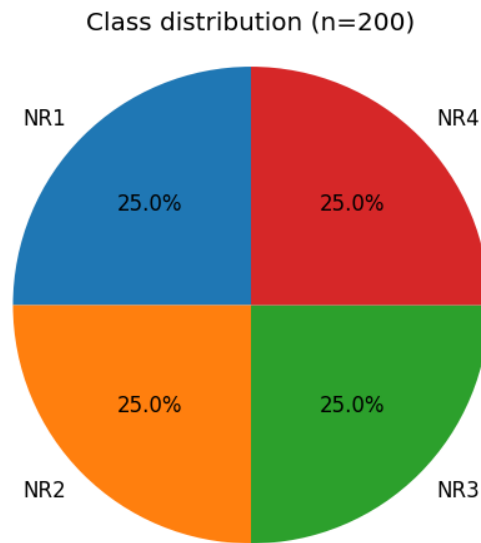


Figure 5. Pie chart of class counts when labels are provided. Balanced slices indicate even representation, while skewed proportions highlight imbalance that can bias classifiers and require corrective strategies such as weighting, resampling, or stratified evaluation.

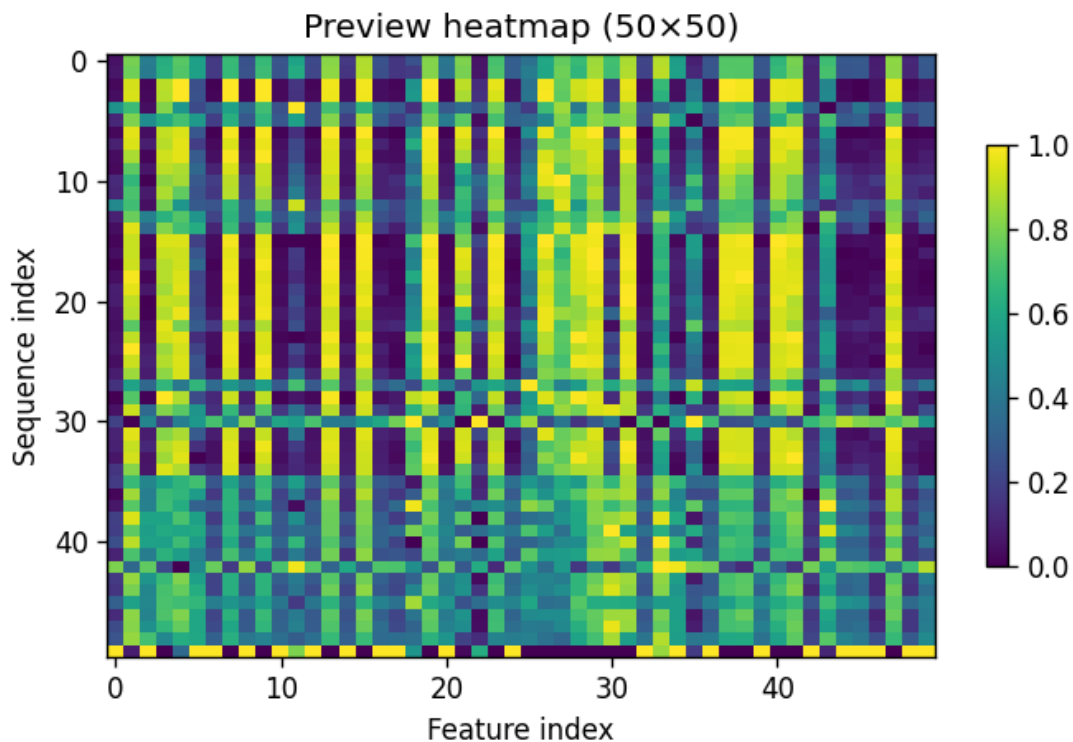


Figure 6. Heatmap preview shows normalized visualization of a small block of the feature matrix. Vertical bands indicate constant or low-variance features, horizontal bands suggest duplicated or

highly similar sequences, and block-like structures reveal correlated features or motif signals. This preview provides an at-a-glance diagnostic of structural patterns in the data.

References

1. Hersh RT. Atlas of Protein Sequence and Structure, 1966. *Syst Biol.* 1967;16(3):262-3. doi:10.2307/2412074.
2. Qian N, Sejnowski TJ. Predicting the secondary structure of globular proteins using neural network models. *J Mol Biol.* 1988;202(4):865-84. doi:10.1016/0022-2836(88)90564-5.
3. Krogh A, Brown M, Mian IS, Sjölander K, Haussler D. Hidden Markov models in computational biology. Applications to protein modeling. *J Mol Biol.* 1994;235(5):1501-31. doi:10.1006/jmbi.1994.1104.
4. Sonnenburg S, Rätsch G, Jagota A, Müller KR. New methods for splice site recognition. In: *Artificial Neural Networks—ICANN 2002*. Berlin, Heidelberg: Springer; 2002. p. 329-36.
5. Sonnenburg S, Schweikert G, Philips P, Behr J, Rätsch G. Accurate splice site prediction using support vector machines. *BMC Bioinformatics.* 2007;8 Suppl 10:S7. doi:10.1186/1471-2105-8-S10-S7.
6. Chen Z, Zhao P, Li F, Leier A, Marquez-Lago TT, Wang Y, et al. iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences. *Bioinformatics.* 2018;34(14):2499-502. doi:10.1093/bioinformatics/bty140.
7. Ismail H, White C, Al-Barakati H, Newman RH, Kc DB. FEPS: A tool for feature extraction from protein sequence. In: Kc DB, editor. *Computational Methods for Predicting Post-Translational Modification Sites*. New York: Springer US; 2022. p. 65-104. doi:10.1007/978-1-0716-2317-6_3.
8. Alipanahi B, DeLong A, Weirauch MT, Frey BJ. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat Biotechnol.* 2015;33(8):831-8. doi:10.1038/nbt.3300.
9. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat Methods.* 2015;12(10):931-4. doi:10.1038/nmeth.3547.
10. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. *Nature.* 2021;596(7873):583-9. doi:10.1038/s41586-021-03819-2.
11. Asgari E, Mofrad MR. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS One.* 2015;10(11):e0141287.
12. Kimothi D, Soni A, Biyani P, Hogan JM. Distributed representations for biological sequence analysis. *arXiv.* 2016. doi:10.48550/arXiv.1608.05949.
13. Min S, Lee B, Yoon S. Deep learning in bioinformatics. *Brief Bioinform.* 2016;18(5):851-69. doi:10.1093/bib/bbw068.
14. Tan J, Hammond JH, Hogan DA, Greene CS. ADAGE-based integration of publicly available *Pseudomonas aeruginosa* gene expression data with denoising autoencoders illuminates microbe-host interactions. *mSystems.* 2016;1(1):e00025-15. doi:10.1128/mSystems.00025-15.
15. Grover A, Leskovec J. node2vec: Scalable feature learning for networks. *Proc 22nd ACM SIGKDD Int Conf Knowl Discov Data Min.* 2016:855-64. doi:10.1145/2939672.2939754.
16. Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S. graph2vec: Learning distributed representations of graphs. *arXiv.* 2017. doi:10.48550/arXiv.1707.05005.

17. Ji Y, Zhou Z, Liu H, Davuluri RV. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*. 2021;37(15):2112-20.
18. Elnaggar A, Heinzinger M, Dallago C, Rehawi G, Wang Y, Jones L, et al. ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans Pattern Anal Mach Intell*. 2021;44(10):7112-27.
19. Lin Z, Akin H, Rao R, Hie B, Zhu Z, Lu W, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*. 2023;379(6637):1123-30.
20. Ismail H, Smith M, Kc D. FEPS: Feature Extraction from Protein Sequences webserver. 2016.
21. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *arXiv*. 2013. doi:10.48550/arXiv.1301.3781.
22. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *Trans Assoc Comput Linguist*. 2017;5:135-46.
23. Ng P. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv*. 2017. doi:10.48550/arXiv.1701.06279.
24. Le Q, Mikolov T. Distributed representations of sentences and documents. *Proc Int Conf Mach Learn*. 2014:1188-96.
25. Yang B, Liu F, Ren C, Ouyang Z, Xie Z, Bo X, et al. BiRen: predicting enhancers with a deep-learning-based model using the DNA sequence alone. *Bioinformatics*. 2017;33(13):1930-6.
26. Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat Commun*. 2019;10(1):390.
27. Zeng H, Edwards MD, Liu G, Gifford DK. Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*. 2016;32(12):i121-7.
28. Singh R, Lanchantin J, Robins G, Qi Y. DeepChrome: deep-learning for predicting gene expression from histone modifications. *Bioinformatics*. 2016;32(17):i639-48.
29. Hamilton WL, Ying R, Leskovec J. Representation learning on graphs: Methods and applications. *arXiv*. 2017. doi:10.48550/arXiv.1709.05584.
30. Jha K, Saha S, Singh H. Prediction of protein–protein interaction using graph neural networks. *Sci Rep*. 2022;12(1):8360. doi:10.1038/s41598-022-12201-9.
31. Fout A, Byrd J, Shariat B, Ben-Hur A. Protein interface prediction using graph convolutional networks. *Adv Neural Inf Process Syst*. 2017;30:6530-9.
32. Rives A, Meier J, Sercu T, Goyal S, Lin Z, Liu J, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc Natl Acad Sci U S A*. 2021;118(15):e2016239118.
33. Pakhrin SC, Pokharel S, Pratyush P, Chaudhari M, Ismail HD, Kc DB. LMPhosSite: A deep learning-based approach for general protein phosphorylation site prediction using embeddings from the local window sequence and pretrained protein language model. *J Proteome Res*. 2023;22(8):2548-57. doi:10.1021/acs.jproteome.2c00667.
34. Pokharel S, Pratyush P, Ismail HD, Ma J, Kc DB. Integrating embeddings from multiple protein language models to improve protein O-GlcNAc site prediction. *Int J Mol Sci*. 2023;24(21):16000.
35. Pratyush P, Bahmani S, Pokharel S, Ismail HD, Kc DB. LMCrot: An enhanced protein crotonylation site predictor by leveraging an interpretable window-level embedding from a transformer-based protein language model. *Bioinformatics*. 2024;40(5):btac290.

36. Pratyush P, Carrier C, Pokharel S, Ismail HD, Chaudhari M, Kc DB. CaLMPhosKAN: Prediction of general phosphorylation sites in proteins via fusion of codon-aware embeddings with amino acid-aware embeddings and wavelet-based Kolmogorov–Arnold network. *Bioinformatics*. 2025;41(4):btaf124.