**Classification of West Nile Virus Genomes using HyenaDNA**

**Background and principles:** Introduction to WNV classification and HyenaDNA technology.

**Data collection and preparation:** Retrieving and preprocessing WNV genomic sequences from GenBank.

**HyenaDNA analysis:** Processing sequences with HyenaDNA and extracting features.

**Classification and validation:** Classifying WNV lineages and evaluating model performance.

**Interpretation and reporting:** Analyzing results and creating phylogenetic trees.

**Comprehensive Procedure for Classifying West Nile Virus Genomes in GenBank Using HyenaDNA**

**1 Background and Principles of WNV Classification and HyenaDNA Technology**

West Nile Virus (WNV) is an enveloped, single-stranded RNA virus belonging to the Flaviviridae family and is transmitted primarily through mosquito vectors, especially of the Culex species 1. First identified in Uganda in 1937, WNV has since spread globally, with significant outbreaks occurring across Africa, Europe, Asia, and North America 14. The virus demonstrates considerable genetic diversity, classified into multiple lineages, with lineages 1 and 2 being the most clinically relevant and widely distributed 24. Accurate classification of WNV genomes is crucial for understanding its epidemiology, evolution, and spread patterns, particularly given its public health impact and potential to cause severe neurological disease in humans and animals 15.

HyenaDNA represents a cutting-edge approach in bioinformatics that combines long-range sequence modeling with efficient computational methods to handle genomic data. Unlike traditional methods that rely on multiple sequence alignments or k-mer based approaches, HyenaDNA uses advanced neural network architectures to capture both local and global sequence contexts across entire viral genomes. This technology is particularly suited for WNV classification because it can: (1) process full-length genomic sequences (approximately 11 kb) without fragmentation; (2) identify subtle genetic patterns that might distinguish different lineages; and (3) incorporate evolutionary relationships through attention mechanisms that span large genetic distances. This approach enables more accurate classification of WNV strains into their respective lineages and sub-lineages, which is essential for tracking geographic spread and understanding viral evolution.

**2 Data Collection and Preparation**

**2.1 Retrieving WNV Genomic Sequences from GenBank**

The first step in the classification process involves gathering a comprehensive set of WNV genomic sequences from GenBank:

Search Query Formulation: Construct targeted search queries using appropriate keywords and filters. Example queries include:

"West Nile virus"[Organism] AND ("complete genome"[All Fields] OR "whole genome"[All Fields])

"WNV"[All Fields] AND ("complete genome"[All Fields]) AND ("lineage 1"[All Fields] OR "lineage 2"[All Fields])

Specific accession numbers for reference sequences (e.g., AF196835.2 for lineage 1, DQ116961.1 for lineage 2) 2

Metadata Collection: For each sequence, retrieve accompanying metadata that will be essential for validation and interpretation:

Collection date and geographical location (country/region)

Host source (human, bird, mosquito, tick)

Lineage designation if already annotated in GenBank

Viral strain name and isolation history

**Data Export:** Download sequences in FASTA format and corresponding metadata in CSV or XML format. The number of sequences should be sufficient to represent the genetic diversity of WNV, ideally including hundreds to thousands of sequences across multiple lineages.

**2.2 Sequence Preprocessing and Quality Control**

Before analysis, sequences must undergo rigorous quality control measures:

Sequence Trimming: Remove any non-viral or vector sequences that might have been included during the sequencing process using tools like BBDuk or Trimmomatic.

Quality Assessment: Check sequences for completeness and integrity:

Ensure sequences are approximately 11,000 nucleotides in length (characteristic of complete WNV genomes) 23

Verify the presence of conserved regions and open reading frame integrity

Check for excessive ambiguous bases (N's) – sequences with >5% ambiguous bases should be excluded.

Redundancy Reduction: Use CD-HIT-EST to cluster sequences at 99% identity to avoid overrepresentation of highly similar sequences, which can bias the classification model.

Multiple Sequence Alignment: Perform alignment using MAFFT or MUSCLE to identify conserved regions and verify sequence orientation. This step, while not directly used by HyenaDNA, helps in visualizing the genetic diversity and ensuring data quality.

**Table: Essential Metadata for WNV Classification**

| Field Name | Description | Example Values |
| --- | --- | --- |
| Accession Number | Unique GenBank identifier | MZ605381, MZ605382 |
| Collection Date | Date of sample collection | 2024-09-07 |
| Country | Origin of sample | Portugal, USA, Kenya |
| Host | Source of the virus | Human, Bird, Mosquito |
| Lineage | Known lineage classification | 1a, 2, 1b |
| Length | Sequence length | 11025, 11028 |

**3 HyenaDNA Analysis Procedure**

**3.1 Installation and Setup of HyenaDNA**

Implementing HyenaDNA requires specific computational environment setup:

Hardware Requirements:

GPU-enabled system (NVIDIA GPU with ≥8GB VRAM recommended for large batches)

≥16GB RAM for processing full genomes

Storage capacity for thousands of viral genomes (relatively minimal due to compressed formats)

Software Dependencies:

Python 3.8+ with PyTorch 1.12+

HuggingFace Transformers library

BioPython for sequence handling

Standard scientific computing packages (NumPy, Pandas, Scikit-learn)

HyenaDNA Implementation:

python

# Example installation commands

!pip install transformers

!pip install biopython

!pip install numpy pandas scikit-learn

# Import necessary librariesfrom transformers import HyenaDNAModel, HyenaDNAConfigfrom Bio import SeqIOimport torchimport numpy as np

**3.2 Sequence Processing with HyenaDNA**

The core analysis involves processing WNV sequences through the HyenaDNA model:

Sequence Encoding: Convert nucleotide sequences to integer tokens using appropriate tokenization:

python

· # Define tokenization dictionary

token\_dict = {'A': 0, 'C': 1, 'G': 2, 'T': 3, 'N': 4}

def sequence\_to\_tokens(seq):

return [token\_dict.get(base, 4) for base in seq.upper()]

· · Model Configuration: Set up HyenaDNA with parameters appropriate for viral genome analysis:

· python

· · # Initialize model with appropriate parameters

config = HyenaDNAConfig(

vocab\_size=5, # A, C, G, T, N

max\_position\_embeddings=11000, # Adjusted for WNV length

num\_attention\_heads=8,

hidden\_size=256,

num\_hidden\_layers=6)

model = HyenaDNAModel(config)

· · Feature Extraction: Process sequences through the model to obtain meaningful representations:

· python

· · # Example feature extraction codedef extract\_features(sequences, model):

model.eval()

all\_features = []

with torch.no\_grad():

for seq in sequences:

tokens = torch.tensor(sequence\_to\_tokens(seq)).unsqueeze(0)

outputs = model(tokens)

features = outputs.last\_hidden\_state.mean(dim=1).squeeze()

all\_features.append(features.numpy())

return np.array(all\_features)

· · Batch Processing: For large datasets, implement batch processing to efficiently handle memory constraints:

· python

·

# Batch processing implementation

batch\_size = 32

features\_batches = []

for i in range(0, len(sequences), batch\_size):

batch = sequences[i:i+batch\_size]

features\_batch = extract\_features(batch, model)

features\_batches.append(features\_batch)

all\_features = np.vstack(features\_batches)

**4 Classification and Validation**

4.1 Dimensionality Reduction and Cluster Analysis

Before classification, reduce the dimensionality of HyenaDNA features to visualize and validate the clustering patterns:

Principal Component Analysis (PCA): Apply PCA to the high-dimensional feature vectors to visualize clusters in 2D or 3D space. This helps in assessing whether sequences naturally group by lineage without explicit supervision.

t-SNE Visualization: Use t-Distributed Stochastic Neighbor Embedding (t-SNE) for nonlinear dimensionality reduction, which often reveals finer cluster structures that might correspond to sub-lineages or geographic groupings.

UMAP Projection: Implement Uniform Manifold Approximation and Projection (UMAP) as an alternative to t-SNE, which often preserves more global structure while still revealing local clusters.

4.2 Building the Classification Model

Construct a machine learning pipeline to classify WNV sequences into lineages:

Model Selection: Choose appropriate classifiers based on the dataset size and complexity:

Support Vector Machines (SVM): Effective for high-dimensional data with clear margins between classes

Random Forest: Handles nonlinear relationships and provides feature importance metrics

Gradient Boosting Machines (XGBoost, LightGBM): Often provides state-of-the-art performance for structured data

Neural Network Classifier: Deep learning approach that can capture complex patterns in the features

**Model Training:**

python

· from sklearn.svm import SVCfrom sklearn.model\_selection import train\_test\_splitfrom sklearn.metrics import classification\_report, confusion\_matrix

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

all\_features, lineages, test\_size=0.2, random\_state=42, stratify=lineages)

# Train SVM classifier

svm\_model = SVC(kernel='rbf', C=10, gamma='scale')

svm\_model.fit(X\_train, y\_train)

# Evaluate performance

y\_pred = svm\_model.predict(X\_test)print(classification\_report(y\_test, y\_pred))

· · Cross-Validation: Implement k-fold cross-validation to ensure robustness of the classification model and avoid overfitting:

· python

·

from sklearn.model\_selection import StratifiedKFold, cross\_val\_score

cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_scores = cross\_val\_score(svm\_model, all\_features, lineages, cv=cv, scoring='accuracy')print(f"Cross-validation accuracy: {np.mean(cv\_scores):.3f} (±{np.std(cv\_scores):.3f})")

**4.3 Validation Against Known Lineages**

Validate the classification results using established reference sequences:

Reference Dataset: Include well-characterized reference sequences from different lineages:

Lineage 1a: NY99 strain (AF196835.2) 2

Lineage 2: Héja strain (MZ605382) 2

Other lineages as available (e.g., lineage 3 Rabensberg, lineage 4 Russia, lineage 5 India, lineage 6 Koutango) 4

**Phylogenetic Validation:** Build a phylogenetic tree using traditional methods (Maximum Likelihood or Bayesian inference) for comparison with HyenaDNA classification results. Use software like IQ-TREE or RAxML with appropriate substitution models for RNA viruses.

**Discordance Analysis:** Investigate any sequences where HyenaDNA classification contradicts existing lineage designations. This may indicate:

Novel genetic variants or potential recombination events

Mislabeled sequences in public databases

Limitations in the current classification model that need addressing

**Table: Performance Metrics for WNV Lineage Classification**

| Metric | Lineage 1a | Lineage 2 | Other Lineages | Overall |
| --- | --- | --- | --- | --- |
| Precision |  |  |  |  |
| Recall |  |  |  |  |
| F1-Score |  |  |  |  |
| Support (# sequences) |  |  |  |  |

**5 Interpretation and Reporting**

**5.1 Results Analysis and Visualization**

Effectively communicate the classification results through comprehensive visualization:

Confusion Matrix: Create a heatmap-style confusion matrix to visualize classification performance across different lineages, highlighting any consistent misclassifications that might indicate genetic similarity between lineages.

Feature Importance: Analyze which genomic regions contribute most to classification decisions by using attention weights or gradient-based importance methods. This can reveal biologically meaningful patterns, such as conserved regions or lineage-specific mutations.

Geographic Mapping: Plot the geographic distribution of different lineages using the metadata associated with sequences, which can reveal patterns of virus spread and migration routes 5.

**5.2 Phylogenetic Tree Construction**

Complement the HyenaDNA classification with phylogenetic analysis:

Multiple Sequence Alignment: Perform rigorous alignment of all sequences using MAFFT or MUSCLE with parameters optimized for flaviviruses.

Tree Building: Implement both distance-based (Neighbor-Joining) and character-based (Maximum Likelihood, Bayesian Inference) methods to reconstruct phylogenetic relationships.

Tree Visualization: Use tools like FigTree or iTOL to create publication-quality trees that incorporate lineage classifications from HyenaDNA as additional annotations.

5.3 Reporting and Database Submission

Generate comprehensive reports and contribute to public knowledge:

Classification Report: Create a detailed document that includes:

Methods summary with parameters and software versions

Complete classification results for all sequences analyzed

Discordance analysis and resolution of conflicting classifications

Novel lineages or genetic variants identified

GenBank Annotation: Submit updated lineage classifications to GenBank using the BankIt tool or Table2asn, providing evidence from both HyenaDNA and traditional phylogenetic analysis.

Public Dashboard: Develop an interactive web-based dashboard that allows researchers to explore the classification results, filter by various criteria, and download subsets of data for their own research.

**6 Conclusion and Future Directions**

The application of HyenaDNA for West Nile virus genome classification represents a significant advancement over traditional methods, offering faster processing times and the ability to detect subtle patterns that might be missed by conventional approaches. This methodology can be readily adapted for other RNA viruses with similar genomic structures and classification needs, such as dengue virus, Zika virus, or influenza viruses.

**Future improvements to the pipeline could include:**

Incorporating temporal data to track the evolution of lineages over time

*Integration with geographic information systems to visualize spatial spread patterns*

Extension to metagenomic data for direct classification from sequencing reads without prior assembly

Development of real-time classification systems for outbreak surveillance and response

By implementing this comprehensive procedure, researchers can efficiently classify West Nile virus genomes from GenBank, contributing to our understanding of its evolution and spread, and ultimately supporting public health efforts to monitor and control this significant pathogen.

**References**

NCBI Bookshelf. West Nile Virus - StatPearls 1

Kreß J, et al. Genome Sequences of West Nile Virus Reference Materials 2

Silva JR, et al. First Isolation and Genome Sequence Analysis of West Nile Virus in Mosquitoes in Brazil 3

Lutomiah J, et al. Whole genome phylogenetic investigation of a West Nile virus strain isolated from a tick sampled from livestock in north eastern Kenya 4

Barros SC, et al. Tracking the Pathways of West Nile Virus: Phylogenetic and Phylogeographic Analysis of a 2024 Isolate from Portugal 5