Transformer-Based Classification of Transposable Element Consensus Sequences with TEclass2

Lucas Bickmann , Matias Rodriguez , Xiaoyi Jiang , Wojciech Makalowski *

Posted Date: 20 November 2025

doi: 10.20944/preprints202511.1598.v1

Keywords: transposable elements; TE classification; transformer architecture; deep learning; bioinformatics web tool

 doi:10.20944/preprints202511.1598.v1

Article

Transformer-Based Classification of Transposable Element Consensus Sequences with TEclass2

Lucas Bickmann [1], Matias Rodriguez [2], Xiaoyi Jiang [1] and Wojciech Makałowski [2,3,*]

[1] Department of Computer Science, University of Münster, Münster, Germany
[2] Institute of Bioinformatics, Faculty of Medicine, University of Münster, Münster, Germany
[3] Department of Genetics, Adam Mickiewicz University, Poznań, Poland
* Correspondence: wojmak@uni-muenster.de

Simple Summary

Transposable elements (TEs) are mobile genetic elements that are present in great numbers in the majority of eukaryotic genomes. They are major drivers of genome evolution as they can facilitate chromosome rearrangements, provide mechanisms for genomic shuffling, and contribute to genome expansion, thereby altering genome architecture. Additionally, TEs can also modify gene expression by disrupting regulatory sequences, impairing genes, and promoting the emergence of new sequences. Despite their abundance, identifying TEs is challenging and time-consuming due to their extreme diversity in DNA sequence. Many TE families are ancient, and most of their sequences have become inactive due to accumulated mutations and fragmentation. Consequently, different copies of the same TE can differ greatly from each other, which makes identifying decayed copies particularly difficult. In this work, we employ Transformer architecture for TE classification. TEclass2 is an integrated classifier that rapidly predicts TE orders and superfamilies using models built on this advanced machine learning approach. The software is available through a web interface, allowing users to classify sequences into sixteen superfamilies according to the Wicker classification system, or alternatively, users can download the source code to train and build custom classification models.

Abstract

Transposable elements (TEs) constitute a significant portion of eukaryotic genomes and play crucial roles in genome evolution, yet their diverse and complex sequences pose challenges for accurate classification. Existing tools often lack reliability in TE classification, limiting genomic analyses. Here we present TEclass2, a software employing a deep learning approach based on a linear Transformer architecture with k-mer tokenization and sequence-specific adaptations to classify TE consensus sequences into sixteen superfamilies. TEclass2 demonstrates improved classification performance and offers flexible model training on custom datasets. Accessible via a web interface with pre-trained models, TEclass2 facilitates rapid and reliable TE classification. These advancements provide a foundation for enhanced genomic annotation and support further bioinformatics research involving transposable elements.

Keywords: transposable elements; TE classification; transformer architecture; deep learning; bioinformatics web tool

 doi:10.20944/preprints202511.1598.v1

1. Introduction

Transposable elements (TEs) are mobile genetic elements which are present in great numbers in the majority of eukaryotic genomes. For example, TEs constitute up to 46% of the human genome [1] and as much as 85% of the genomes of some plants, such as wheat and maize [2,3]. TEs are major drivers of genome evolution as they can facilitate chromosome rearrangements by acting as recombination hotspots, they can also provide mechanisms for genomic shuffling [4], and contribute to genome expansion [5,6], thereby altering genome architecture. Additionally, TEs can modify gene expression by disrupting regulatory sequences, impairing genes, and promoting the emergence of new sequences [7].

Despite their abundance, identifying TEs remains challenging and time-consuming because they are extremely diverse in DNA sequence, exhibiting a wide range of motifs, lengths, and structures. Many TE families are ancient, and most of their sequences have become inactive due to accumulated mutations and fragmentation. As a result, different copies of the same TE can differ greatly from each other, making the identification of decayed copies particularly difficult [8].

There are two main strategies for identifying TEs. The first involves performing similarity searches against the genome using libraries of known TE sequences. The second, particularly useful for non-model organisms or for discovering novel TEs, is a de novo approach. In this method, interspersed repetitive sequences are identified directly from the genome and used to build consensus models of candidate TEs, which are then compared to known TE libraries for classification and annotation [9].

Multiple approaches can be used for the classification of TEs depending on structural features, transposition mechanisms, enzymatic machinery or the evolutionary context [10]. The first proposal for classifying TEs [11] divided them into two classes based on their mechanism of transposition. The TEs that use the reverse transcription of an RNA intermediate are called class I elements or retrotransposons, while the TEs that transpose using a DNA intermediate are called class II elements or DNA transposons. Class I TEs were further divided into two categories, LTR-retrotransposons and non-LTR retrotransposons. LTR-retrotransposons are similar to retroviruses, with open reading frames flanked by long terminal repeats (LTR), whereas non-LTR retrotransposons have no LTRs or very short terminal repeats and a poly-A tail.

This initial classification did not account for non-autonomous elements such as SINEs, and has since been revised and updated multiple times. Two of the most widely used classification systems are those from Repbase [12,13] and Wicker et al. [14]. Both systems introduce additional subdivisions to the original two TE classes, considering factors such as sequence similarity, structural features, and the number of open reading frames (ORFs) and encoded enzymes. The Repbase classification divides TEs into three main classes: DNA transposons, LTR retrotransposons, and non-LTR retrotransposons, with further subdivisions resulting in 65 superfamilies or clades [15]. In contrast, the Wicker system divides class I elements into five orders (LTR, DIRS, PLE, LINEs, and SINEs), and class II elements into two subclasses: subclass 1 (cut-and-paste TEs, including TIR and Crypton elements) and subclass 2 (copy-and-paste TEs, including Helitron and Maverick elements) [14].

The need to identify and classify these large genomic components has led to the development of pipelines for semi-automated TE discovery and annotation. Most commonly used tools classify TEs by comparing consensus sequences to TE databases based on sequence similarity. For example, RepeatModeler uses the DFAM database for TE classification, while REPET uses its own RepetDB, which has a classification hierarchy similar to Repbase. Alternative approaches to sequence homology include tools like PASTEC [16], part of the REPET pipeline, which uses hidden Markov model (HMM) profiles to identify TEs.

Another promising strategy is the use of machine learning, which has shown significant potential in bioinformatics [17]. Its application has become increasingly reliable and feasible due to the rapid growth of genomic data and advances in deep learning algorithms that have improved both speed and accuracy [18,19]. Machine learning techniques can automatically identify patterns and extract information from data, making them well-suited for classifying sequences such as DNA.

 doi:10.20944/preprints202511.1598.v1

The first software to use machine learning for classifying TEs was TEclass [20], which used support vector machines (SVMs) to classify unknown TEs based on the frequencies of tetramers and pentamers in repeat elements. TEs were classified into main categories: DNA transposons, LTRs, non-LTR repeats, and further into LINEs or SINEs. The classifiers, built using TE sequences from Repbase, employed a stepwise binary classification: first distinguishing DNA transposons from retrotransposons, then separating LTRs from non-LTRs among retroelements, and finally distinguishing LINEs from SINEs among non-LTR elements [20].

In recent years, deep learning methods based on neural networks have enabled the development of new tools for TE classification. For example, DeepTE [21], TERL [22], and Terrier [23] use convolutional neural network (CNN) architectures to classify TEs into their respective orders or superfamilies. Some tools are specialized for specific TE orders: TIR-Learner [24] classifies TIR elements using neural networks, k-nearest neighbor, random forest, and Adaboost; TE-Learner [25] uses a random forest approach for LTR elements; and Inpactor2 [26] applies CNNs to classify LTR-retrotransposons in plant genomes. In TEclass2, we employ the Transformer deep learning model [27], t originally developed for natural language processing (NLP) and computer vision. Transformers have demonstrated exceptional performance in these fields, often surpassing models such as recurrent neural networks (RNNs) [28]. They process entire inputs simultaneously using an attention mechanism that provides context for any position in a sequence, enabling parallelization and reducing training time [27].

Other studies have also applied Transformers in bioinformatics, such as DNABERT [29], which is based on the BERT Transformer language model [30]. and used for predicting promoters, splice sites, and transcription factor binding sites. Similar Transformer-based approaches have been used for bacterial promoter classification, viral genome identification, and analysis of mRNA degradation properties in COVID-19 vaccine candidates [31]. Recently, InstaDeep group developed series of Tranformer-based models for human genomics [32].

While vanilla Transformer and BERT models achieve state-of-the-art performance in their respective fields, they have a runtime and space complexity of $O(n^2)$, typically limiting them to processing sequences of up to 512 tokens. To address this, linear Transformer models such as Linformer [33] and Longformer [34] have been developed, allowing processing of sequences up to 4096 tokens or more, depending on available memory. However, these architectures are not yet fully adapted for continuous, unseparated sequences like those found in DNA.

In this work, we introduce a new architecture based on the Longformer model [34] for classifying selected TE sequences, incorporating sequence-specific augmentations, a k-mer specialized tokenizer, and sliding window dilation. TEclass2 is an integrated classifier that rapidly predicts TE orders and superfamilies using models built on this advanced Transformer architecture. The software is available through a web interface, allowing users to classify sequences into sixteen superfamilies according to the Wicker classification system [14], or alternatively, users can download the source code to train and build custom classification models.

2. Methods

2.1. Transformer

A Transformer is a deep learning model that uses the mechanism of self-attention originally designed to handle sequential input, especially natural language, and is vastly improved in its performance when compared to its predecessors, the Recurrent Neural Networks (RNNs) [35] and Long Short-Term Memory (LSTM) [36]. Instead of using the input as a sequence and processing each word in order as RNNs do, they process the whole sentence concurrently, allowing parallel computation and improving long-term dependencies. Originally transformers were designed using sequence transformation with a Seq2Seq model for performing tasks like translation and summarization of text. This model has an encoder that captures the input and passes it to a decoder that produces an output [37].

 doi:10.20944/preprints202511.1598.v1

In TEclass2, for the classification of TE DNA sequences, we use only the encoder-block, followed by a classification head as in a linear layer (Figure 1). The transformer model is divided into several layers which have multiple attention heads [38] and each one that can learn independently relevant parts of the input sequence.

The first layer's query is the encoded query or tokens and the following layers are based upon previously computed attention matrices as queries. In order to take into consideration the order of the input sequence, we added positional encoding. Due to memory constraints transformers can process up to 512 tokens and larger inputs are either truncated or processed in chunks. For the latter the classification is then computed as the median of the output from each of the processed chunks. To solve this problem, different methods have been proposed such as the Longformer method which tackles this with reduced self-attention masks of local attentions and by adding some global-attentions [34].

A way to increase variants in the data during training is to use data augmentation and forcing the machine learning model to adapt to more generalized features [39]. To apply an on-the-fly sequence augmentation during the training step, we developed a variety of biologically motivated data augmentations (Table 1).

Table 1. List of data augmentation used in DNA sequences for the data-training step in TEclass2.

Augmentation	Description
SNP	Replace randomly a single nucleotide.
Masking	Replace a base with an N.
Insertion	Insert random nucleotides in a sequence.
Deletion	Delete a random part of a sequence.
Repeat	Repeats a random part of the sequence.
Reverse	Reverses the sequence.
Complement	Computes the complement sequence.
Reverse complement	Computes the opposite strand.
Add tail	Adds poly-A tail to the sequence.
Remove tail	Removes poly-A tail if present.

Depending on the applied augmentations, the DNA sequence may also differ in length. The only transformation that introduces ambiguity is the introduction of Ns in the sequence and in this case the transformer should learn that these characters do not represent relevant information. The original transformer does not handle information about the position of tokens due to the transformers parallel processing of the data, therefore the relationship between tokens has to be provided, as these distances provide important spatial information. Sinusoidal positional encoding is used for this as it has the advantage of providing information that is independent from the length of the input and each value ranges between zero and one [40].

In TEclass2 the DNA sequences used as an input undergo a process known as tokenization, in which the input strings are split into smaller parts called tokens which are assigned identifiers based on a created vocabulary. Tokenizers can also be embedded into a vector space and used as a form of compression and dictionary coding. We used k-mer tokenization with a sliding window approach dividing each DNA sequence into k-mers, whereas the general length of the embedded string does not decrease when using a different window size. A larger window size dilates the steps between the k-mers, which increases the coverage of the input sequence but reduces the overlap and dependencies

 doi:10.20944/preprints202511.1598.v1

between them, as they also have the advantage of generating a fixed number of words. In our software we used k-mers with a length of five with a sliding step of two nucleotides. The number of DNA 5-mers including unknown nucleotides as N gives a rather small vocabulary of 3125 tokens. With this software it is possible to use k-mers of different length but it's important to consider that using the shorter ones impedes the encoding of different semantic representations, and using the larger ones increases the size of the vocabulary manifold degrading the performance [41].

In order to allow long DNA sequences we introduced a dynamically scaling sliding step of minimal two nucleotides to maximize the coverage of the input sequence, while trying to retain overlapping tokens. We compute it as:

$$step\_size = min\left(\left[\frac{len(seq)}{max\_embedding\_size}\right], k-1\right)$$

The attention values are traced back to each token and using linear interpolation these values are scaled back. This reduces cross dependencies between local tokens and increases long-range dependencies, which are useful to classify long TE sequences.

The models tested were trained with weighted cross entropy (WCE), as the datasets are unbalanced. The Softmax function [42] is used to compute the probabilities for predicting the TE classes and to classify the performance of each trained model we used different metrics during evaluation and testing. This includes the train- and evaluation-loss, class-wise precision, recall, F1-score, and global accuracy on the test-dataset; also a macro average and weighted average were computed for precision, recall, and F1-score.

2.2. Workflow

The input data for training and building a model with TEclass2 is a FASTA file with the TE models, where the header of each sequences contains the name and the class of TE. For the training of a model, the workflow (Figure 1) starts by building a database of labeled TE sequences, which is divided into training, evaluation and testing datasets. The configuration of the database is read from the configuration file, where the expected family names and files paths must be specified.

 doi:10.20944/preprints202511.1598.v1

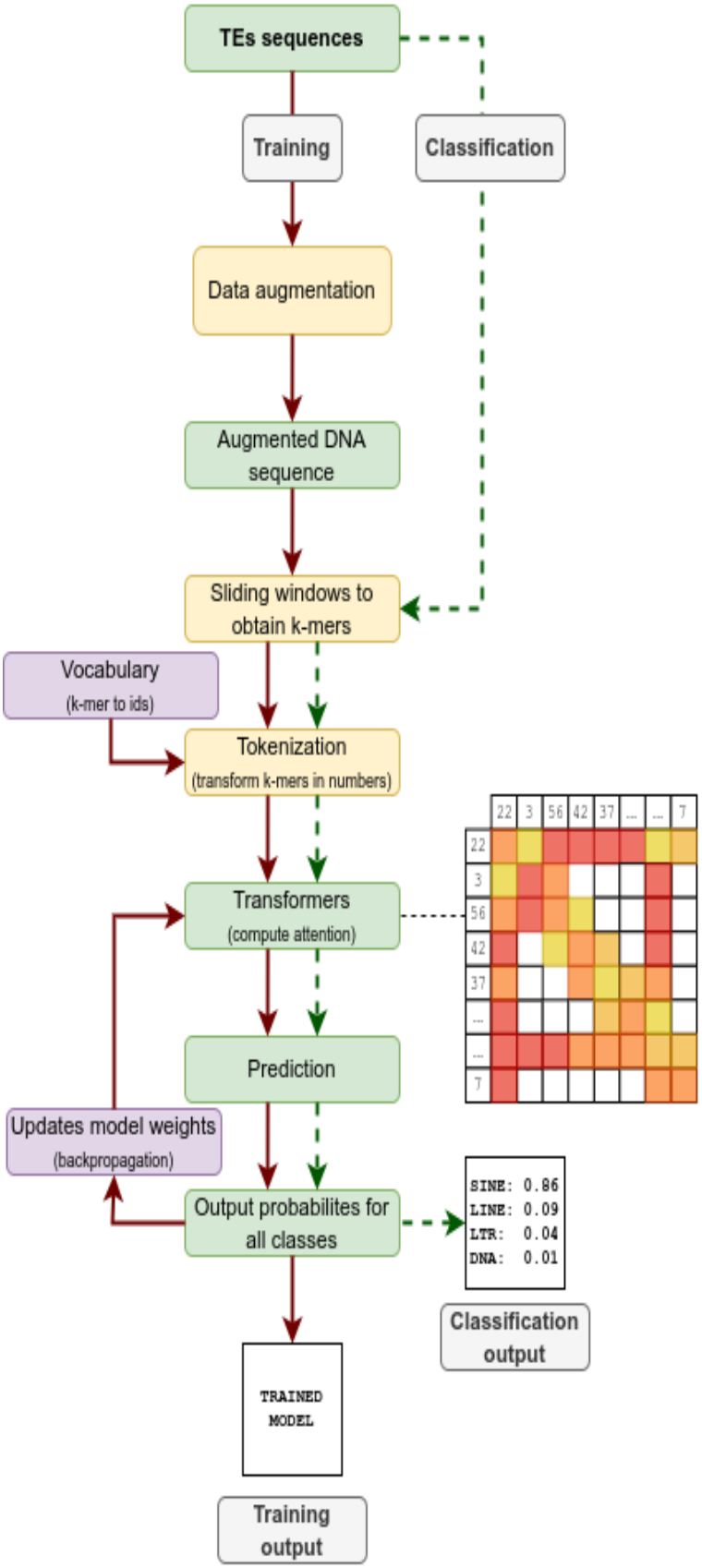


Figure 1. Flowchart succinctly describing how TEclass2 works both in the training of dataset to produce Transformer models (red arrows) and the use of these models to classify TE sequences (dotted green arrows).

 doi:10.20944/preprints202511.1598.v1

After the database has been built, the training step, used to train the model, can begin utilizing parameters from the configuration file such as the number of epochs, the size of the layers for the neural network, size of the embeddings, window dilation, and many others. The training step uses the data augmentation properties specified in Table 1, which increase the variability of the input sequences, then, the TE sequences are scanned using a sliding window approach.

Next, each k-mer is tokenized and used as an input for the transformer algorithm that computes the local attention for the whole sequence and the global attention for certain specific positions. After this step, the TE sequences that were not used for training are evaluated with the Transformer model and a TE class is predicted and compared with the labeled data. During the training a checkpoint is saved after each epoch, which allows the user to restore the training process from a previous step in case the training needs to be restarted or extended. In the final step, the trained model is validated on the test-set, which consists of unseen and unbiased comparable data.

For the classification of TEs, we use sliding windows to compute the k-mers of the input sequence and tokenize these data for use with the Transformer model. The model then outputs a prediction score, which is normalized with the Softmax function and gives the probability of the input sequence of belonging to any of the classes specified in the model.

2.3. Datasets

In the previous version of TEclass, the software permitted the classification of TEs into four categories, namely SINEs, LINEs, LTRs, and DNA transposons. To enhance the classification capabilities and allow a more in-depth categorization of TEs and by taking advantage of the larger datasets available today we built a classification model that assign TEs to sixteen different categories of TEs. These categories are based on the Wicker classification and include eight orders from class I (RNA-mediated transposons), i.e. Copia, Gypsy, Pao, ERV, L1/L2, RTE, jockey, and SINEs and eight orders from class II ("cut-and paste" transposons), i.e. Tc1/Mariner, hAT, Merlin, Transib, P, Crypton, Helitrons, and Mavericks (Polintrons) (see Table 2).

For the creation of the curated dataset for training the models, we used TE sequences from Dfam, curated and non-curated version 3.7, and the curated Repbase version 18. In addition, the curated database was very limited and too small for models that are more complex. To allow more categories of TEs, we need a larger dataset with a larger variety of TE classes. The use of a non-curated database improves the number of data points, albeit with a slight decrease in the quality of the results. From all the TE sequences in the dataset 75% are used for training the model, 15% are used for validation, and 10% for testing (Table 2).

2.4. Evaluation of TE Models

In order to evaluate the performance of the trained models we calculated the precision, recall and F1 score of each one of the TE classes of a model to assess its reliability in the classification. Precision is a measurement of the exactness of the model and is the number of true positives divided by the sum of true positives and false positives. A high value of precision indicates that false positives are rare in the model. Recall is a measurement of the ability of the model to identify all the correct predictions and is defined as the proportion of true positives divided by the sum of true positives and false negatives. A high value of recall indicates a good level of completeness of the classification model, meaning that false negatives are not common. Although both measures are useful, they have their own limitations, as precision does not consider misclassifications meanwhile recall does not consider false positives. The F1 score combines both measures as it is the harmonic mean of precision and recall, where both contribute equally to the score, which has values between 0 and 1.

Table 2. DFAM version 3.7 curated and non-curated data and Repbase version 18 used for training TEclass2, showing how the TEs from different categories were used for training, validation and testing.

Group	Training	Validation	Test	Total

 doi:10.20944/preprints202511.1598.v1

Copia	18584	3717	2478	24779
Crypton	5453	1091	727	7270
ERV	61539	12319	8212	82124
Gypsy	79674	15935	10623	106232
hAT	114707	22941	15294	152943
Helitron	48980	9796	6531	65307
Jockey	14201	2840	1894	18935
L1/L2	113708	22742	15161	151610
Maverick	7218	1444	962	9624
Merlin	2971	594	396	3961
P	4692	938	626	6256
Pao	22192	4438	2959	29589
RTE	43500	8700	5800	58000
SINE	31960	6392	4261	42613
TcMar	86925	17385	11590	115900
Transib	4279	856	571	5705
Total:	**660636**	**132127**	**88085**	**880848**

Since in each model we are using multiple classes with different number of TEs for comparing whole models between themselves, we calculated the F1 weighted average using the following formula:

$$F1_{w-avg} = \sum\left(2\left(\frac{precision * recall}{precision + recall}\right) * \frac{n\_obs\_TE\_class}{n\_total\_TEs}\right)$$

The weighted F1 score takes into consideration the class imbalance of the models and calculates the F1 score for each TE class, which is then multiplied by the proportion of TEs from that class in the total dataset. During the training of the models, these scores are calculated after each training cycle or epoch, as we used 10% from each TE class of the dataset for testing the performance of each model, as shown in the column "Test" from Table 2. In this way, during the training the trajectory of the changes in the performance can be checked in real-time. We also evaluated the performance of the software by classifying TEs models obtained from annotated genomes.

2.5. Software and Hardware Specifications

The hardware used for training were an Nvidia GTX 1660Ti 6 GB, RTX 3080 12 GB, A100 40 GB, and A100 80 GB. Palma II (https://confluence.uni-muenster.de/display/HPC), the HPC of the University of Münster, provided the latter two. These tests were conducted using float32 training, as mixed precision training is still marked as an experimental feature in the Trainer API. Gradient Accumulation was used for models based on non-curated data or very complex parameters, to overcome memory constraints. TEclass2 was developed using Python3 and requires NumPy [43], scikit-learn [44], PyTorch [45], Tensorboard [46], Transformers, and Tokenizer packages [28].

 doi:10.20944/preprints202511.1598.v1

3. Results

3.1. Parameters Used for Training TE Models

During the training of the models, we evaluated different configurations, pre-processing steps, and parameters regarding the complexity of the models. We tested building models using 4-mer and 6-mer, but this resulted in a very similar performance to the default 5-mer models. As expected, allowing longer sequence inputs by increasing the embedding size provides an overall improvement of the results in particular for longer TEs. We noticed the importance of using dilated sliding windows to improve the results when the input contained long sequences. This resulted in an increased F1 score, which impacted the overall performance of the trained model.

Increasing the sparse attention did not improve the precision, indicating that local attention did not have an impact on classification beyond a specific minimal attention window and increasing it did not produce significant improvements. We also noticed that the complexity of each layer beyond a certain point did not increase the performance metrics, but what affected the most was the complexity of the model itself. In contrary to other methods of model complexity, the use of additional global attention slightly increased the overall scores.

To summarize, the complexity of the model and the available data points did not change the performance as much as expected. This introduced a grain fine-tuning with different parameters for TEclass2 (Table 3). The classifier approach using 5-mers with dilated sliding windows, using the maximum embedding size with 2048 tokens alongside a few additional global attention tokens achieved the highest scores.

Table 3. Optimal parameters used to train the Transformer model with both curated and non-curated dataset. These parameters may not be optimal for other datasets and their values depend on the hardware resources available.

Parameters	Values	Description
num_epochs	100	Times the model is trained on the same data.
max_embeddings	2048	Maximum length of the input sequence, but can be overcome with the Longformer model.
num_hidden_layers	8	Dimensions of the encoder hidden state for processing the input.
num_attention_heads	8	Number of heads (neurons) used to process input data and make predictions.
global_att_tokens	[0, 256, 512]	Positions of the input where all the sequence is considered
intermediate_size	3078	Dimensionality of the feed-forward layer
kmer_size	5	Length of the words for dividing the input sequence.

3.2. Training of TE Models

We built a Transformer model that allows the classification of TEs in sixteen categories and the performance results with the training dataset can be observed from a confusion matrix (Figure 2) and a table of model testing statistics (Table 4). For classifying TEs beyond just their main order, as the previous version of TEclass does, the data from curated databases was insufficient, so we relied on the Dfam version 3.7 non-curated dataset and Repbase version 18. We acknowledge the risks

 doi:10.20944/preprints202511.1598.v1

involved in using non-curated data, yet we expect that the majority of the sequences will be correctly classified, outweighing biases introduced by those erroneously labeled.

Table 4. Evaluation parameters obtained from curated dataset using sixteen TE classes.

Group	Precision	Recall	F1-score	Support
Copia	0.69	0.60	0.64	3716
Crypton	0.62	0.69	0.65	1090
ERV	0.86	0.91	0.88	12318
Gypsy	0.81	0.71	0.76	15934
hAT	0.84	0.82	0.83	22941
Helitron	0.72	0.81	0.76	9796
Jockey	0.61	0.71	0.66	2840
L1/L2	0.89	0.80	0.84	22741
Maverick	0.56	0.64	0.60	1443
Merlin	0.64	0.70	0.67	594
P	0.35	0.60	0.44	938
Pao	0.71	0.70	0.70	4438
RTE	0.76	0.78	0.77	8700
SINE	0.76	0.80	0.78	6391
TcMar	0.76	0.77	0.76	17385
Transib	0.67	0.75	0.71	855
Accuracy:			0.79	132120
Macro avg:	0.70	0.74	0.72	132120
Weighted avg:	0.79	0.79	0.79	132120

 doi:10.20944/preprints202511.1598.v1

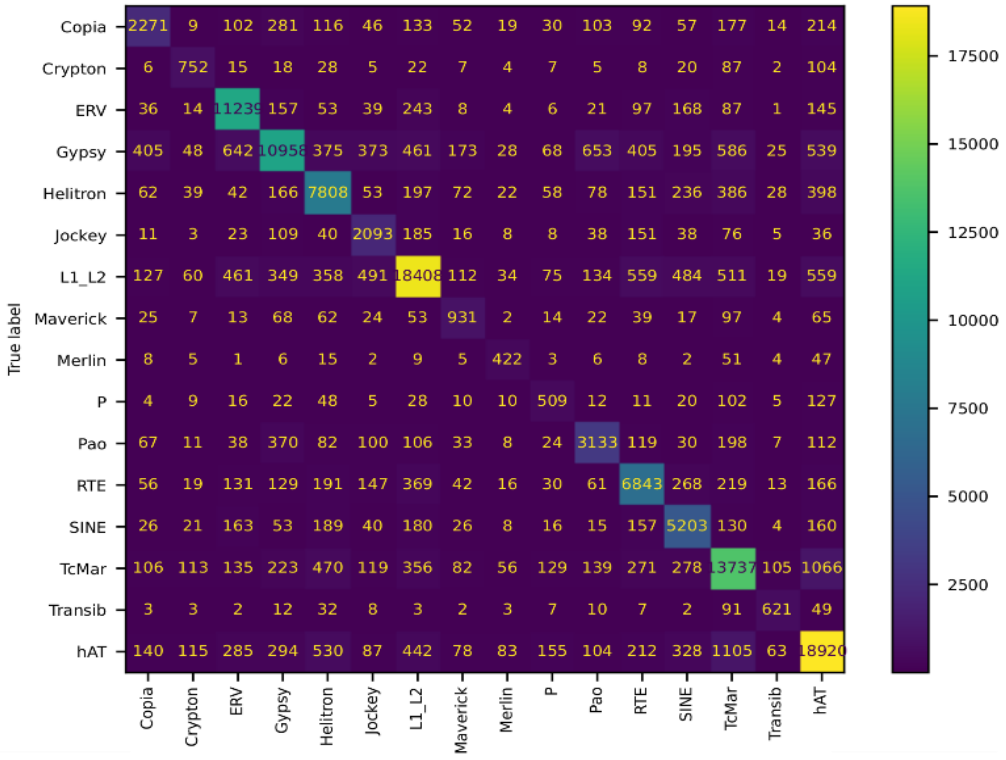


Figure 2. Confusion matrix for the training of non-curated data from Dfam version 3.7 and Repbase version 18.

As expected, when more categories are added in order to train a model, the model increases in complexity, takes more time to train, and there is a greater chance of misidentification. Furthermore, some classes of TEs are similar to each other, making classification difficult. When we train a model using sixteen different classes of TEs, we observe good performance, considering the size of the model for the hAT, L1/L2, and ERV transposons, all obtaining F1-scores greater than 0.8. The performance of the model in classifying the TcMar, SINEs, RTE, Transib, Helitrons, and Gypsy transposons shows F1-scores for all of them greater than 0.7. The performance with some other transposons, like Copia, Crypton, Maverick, Merlin, and P, was suboptimal, with F1-scores below 0.7. In general, the weighted average of the model has an F1-score of 0.76, with a precision score of 0.77 and a recall of 0.76. The misclassification that we observed can be attributed in part to the limitation of the Transformer model itself or to how the models were built, but can also be attributed to mislabeled data in the TEs dataset or to the lack of enough representative sequences for training the model in a given class of TEs.

3.3. Comparison with Other Machine Learning Tools

To assess the classification performance of the TE model trained with TEclass2, we employed TE models built using RepeatModeler2 [47] version 2.0.3 with default parameters and the LTR-detector module on the fruit fly and zebrafish genomes, both obtained from the UCSC Genome Browser database. RepeatModeler already provides TE model classifications through its RepeatClassifier module. To further curate these results, we ran nhmmscan from the HMMER package version 3.3.2 (hmmer.org), with the parameters "--noali -E 1e-10" to align these TE models with the curated Dfam database version 18. Sequences that exhibited poor alignment with TEs or had hits to TEs from different orders were discarded, as this often indicates chimeric models, a common problem in most TE model building software [8].

We ran the TE models obtained from the rice and fruit fly genome with TEclass2 and compared them to the classification assigned by RepeatModeler. We also ran these TE models using the default settings of two other machine learning software, DeepTE and TERL. Making a direct comparison of TE classification tools can be difficult, as some of them specialize in solving specific problems, such as differentiating TEs within the same class or classifying into a different number of families. For the

 doi:10.20944/preprints202511.1598.v1

75 TE models built from the rice genome, most of them are Gypsy and Copia, but are also some Helitrons and LINEs. TEclass2 using a minimum probability threshold of 0.7 was able to classify 79% of the TEs models and from those 82% were identified correctly at superfamily level and for 87% the order was also predicted accurately.

For the fruit-fly genome, 667 TE models were built, with a diverse number of TE models from Gypsy, Jockey, Bel-Pao, R1, Tc1-Mariner, Copia, P, Helitron, Transib, and hAT. From this dataset, 393 were classified with a probability of at least 0.7, that is 59% of the total, and from those 285 were correctly classified, meaning a 73% accuracy (see Supplementary Material). Adopting a stricter probability threshold, such as 0.9, improved the TE prediction accuracy to 83%, but reduced the number of TEs that are classified, to 281 out of 667. During these analyses there was also a small group of TEs that couldn't be classified because they don´t belong to any of the families included in the trained model.

For testing other machine learning tools, we ran DeepTE with the parameters "-m M -sp M" for the fruit fly TEs models in order to specify that TEs are from a Metazoa genome and "-m P -sp P" for rice TEs models to specify that the TEs are coming from a plant genome. DeepTE sets by default a limit of a probability of 0.6 to give a result, if the value is lower, then the TE is not classified. For the TEs models built from the rice genome, the accuracy for predicting the TE order was of 53%, and for the superfamily of 45%. From all the TEs models built, only 79% had their order predicted and 29% their superfamily. For the TE models built from the fruit fly genome, DeepTE was able to predict the order accurately for 35% and the superfamily for 32%. The proportion of TEs with the order predicted was 88% and for the superfamily 58% (See supplementary material).

We also ran TERL for both genomes, using the model DS3, which is a model that can classify TEs into twelve superfamilies. In the case of TE models constructed from the rice genome, the accuracy of predicting the TE order was 68%, and 50% for the superfamily. Among all of the TE models constructed, a classification rate of 91% was achieved as 9% were classified as non-TEs. For TE models from the fruit fly genome, 42% were accurately predicted and 21% for the superfamily. Approximately 57% of the TEs had their order predicted (See supplementary material at https://github.com/IOB-Muenster/TEclass2/tree/main/tests).

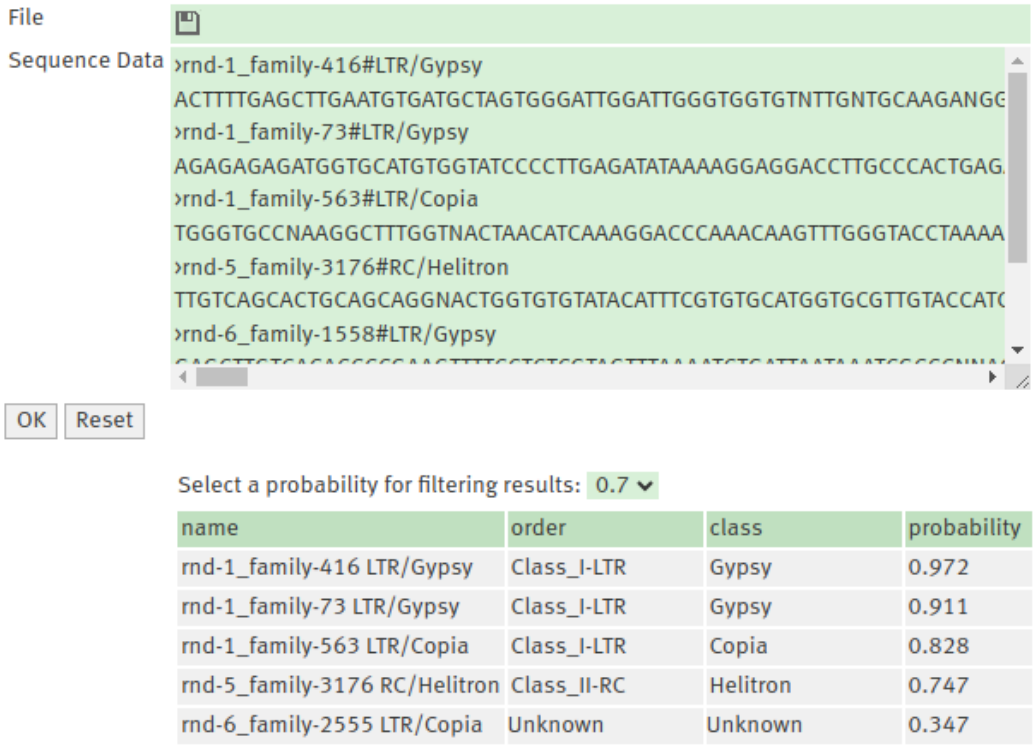
3.4. TEclass2 Web Interface

The models built with curated and non-curated data can be used to classify TEs at: *https://bioinformatics.uni-muenster.de/tools/teclass2/index.pl*

The web interface provides a simple interface that allows the classification of TEs models DNA sequences into sixteen superfamilies and the possibility to filter the results by a probability value. The TE sequences used as input can be uploaded as a FASTA file or inputted in the text box.

The output produced by the classification module provides an accessible and compact format as a table or as a downloadable TSV file useful for further analysis. This table includes the classification predicted, with the order and superfamily that corresponds to the class with the highest probability, and the Softmax confidence scores for each class in the model (Figure 3). For training models it is necessary to download the source code of the program from https://github.com/IOB-Muenster/TEClass2 and follow the configuration instructions in order to run it. The trained models for TEclass2 can be downloaded directly from: https://bioinformatics.uni-muenster.de/tools/teclass2/models/teclass2-models.tgz

 doi:10.20944/preprints202511.1598.v1



name	order	class	probability
rnd-1_family-416 LTR/Gypsy	Class_I-LTR	Gypsy	0.972
rnd-1_family-73 LTR/Gypsy	Class_I-LTR	Gypsy	0.911
rnd-1_family-563 LTR/Copia	Class_I-LTR	Copia	0.828
rnd-5_family-3176 RC/Helitron	Class_II-RC	Helitron	0.747
rnd-6_family-2555 LTR/Copia	Unknown	Unknown	0.347

Figure 3. The web interface for TEclass2 allows inputting the TE sequence to be classified in a text box or as a FASTA file. The output shows the classification and the Softmax values the model scored for each class of TE.

3.5. Known Limitations

Some of the limitations of TEclass2 are the hardware requirements for training new models, as it is necessary to have a GPU to build new models. Other weaknesses come directly from the Transformers architecture [48], as the training of models require a fine-tuning of multiple parameters, which may lead to several trial and error cycles. Inherent to the architecture comes the difficulty to control the attention mechanism, which can put attention in parts of the sequences that are biologically not relevant or show an undesirable attention-bias. The algorithm also needs to fragment the sequences into k-mers meaning that the sequence is not analyzed as a whole but divided into words similarly to NLP inputs.

Another issue may come from the input data itself, as machine-learning strategies require a large number of TE sequences that are well curated. This may be impossible to attain for some classes of TEs where only a few copies are present in curated databases.

4. Conclusions

TEclass2 is a comprehensive classifier that applies a state-of-the-art Transformers architecture and yields promising results. When we trained many TE models, we computed a weighted overall accuracy of 79% for properly classified sequences when dividing TEs into sixteen families. TEclass2 is readily accessible for downloading or online use, providing researchers with a simple and accessible software for classifying unknown TE models.

Author Contributions: Conceptualization, Wojciech Makalowski; Methodology, Matias Rodriguez and Xiaoyi Jiang; Software, Lucas Bickmann and Matias Rodriguez; Validation, Lucas Bickmann and Matias Rodriguez; Investigation, Lucas Bickmann and Matias Rodriguez; Writing – original draft, Lucas Bickmann and Matias Rodriguez; Writing – review & editing, Xiaoyi Jiang and Wojciech Makalowski; Supervision, Xiaoyi Jiang and Wojciech Makalowski; Funding acquisition, Wojciech Makalowski.

 doi:10.20944/preprints202511.1598.v1

Funding: This research was partially funded by the DAAD Research Grants – Doctoral Programmes in Germany 2018/2019 (57381412) to MR and internal funds of the Institute of Bioinformatics.

Acknowledgments: This work utilized the computational resources of the PALMA-II HPC cluster of the University of Münster (https://confluence.uni-muenster.de/display/HPC).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hoyt, S.J., et al., *From telomere to telomere: The transcriptional and epigenetic state of human repeat elements.* Science, 2022. **376**(6588): p. eabk3112.
2. Schnable, P.S., et al., *The B73 Maize Genome: Complexity, Diversity, and Dynamics.* Science, 2009. **326**(5956): p. 1112-1115.
3. Wicker, T., et al., *Impact of transposable elements on genome structure and evolution in bread wheat.* Genome Biol, 2018. **19**(1): p. 103.
4. Moran, J.V., R.J. DeBerardinis, and H.H. Kazazian, Jr., *Exon shuffling by L1 retrotransposition.* Science, 1999. **283**(5407): p. 1530-4.
5. Piegu, B., et al., *Doubling genome size without polyploidization:: Dynamics of retrotransposition-driven genomic expansions in , a wild relative of rice.* Genome Research, 2006. **16**(10): p. 1262-1269.
6. Ungerer, M.C., S.C. Strakosh, and Y. Zhen, *Genome expansion in three hybrid sunflower species is associated with retrotransposon proliferation.* Current Biology, 2006. **16**(20): p. R872-R873.
7. Makalowski, W., *Genomic scrap yard: how genomes utilize all that junk.* Gene, 2000. **259**(1-2): p. 61-7.
8. Rodriguez, M. and W. Makalowski, *Software evaluation for de novo detection of transposons.* Mobile DNA, 2022. **13**(1).
9. Makalowski, W., et al., *Transposable Elements: Classification, Identification, and Their Use As a Tool For Comparative Genomics.* Evolutionary Genomics, 2 Edition, 2019. **1910**: p. 177-207.
10. Piégu, B., et al., *A survey of transposable element classification systems - A call for a fundamental update to meet the challenge of their diversity and complexity.* Molecular Phylogenetics and Evolution, 2015. **86**: p. 90-109.
11. Finnegan, D.J., *Eukaryotic transposable elements and genome evolution.* Trends Genet, 1989. **5**(4): p. 103-7.
12. Jurka, J., et al., *Repbase Update, a database of eukaryotic repetitive elements.* Cytogenet Genome Res, 2005. **110**(1-4): p. 462-7.
13. Kapitonov, V.V. and J. Jurka, *A universal classification of eukaryotic transposable elements implemented in Repbase.* Nat Rev Genet, 2008. **9**(5): p. 411-2; author reply 414.
14. Wicker, T., et al., *A unified classification system for eukaryotic transposable elements.* Nature Reviews Genetics, 2007. **8**(12): p. 973-982.
15. Bao, W., K.K. Kojima, and O. Kohany, *Repbase Update, a database of repetitive elements in eukaryotic genomes.* Mob DNA, 2015. **6**: p. 11.
16. Hoede, C., et al., *PASTEC: an automatic transposable element classification tool.* PLoS One, 2014. **9**(5): p. e91929.
17. Wu, J. and Y.Q. Zhao, *Machine learning technology in the application of genome analysis: A systematic review.* Gene, 2019. **705**: p. 149-156.
18. Lan, K., et al., *A Survey of Data Mining and Deep Learning in Bioinformatics.* J Med Syst, 2018. **42**(8): p. 139.
19. Li, R., et al., *Machine learning meets omics: applications and perspectives.* Brief Bioinform, 2022. **23**(1).
20. Abrusan, G., et al., *TEclass--a tool for automated classification of unknown eukaryotic transposable elements.* Bioinformatics, 2009. **25**(10): p. 1329-30.
21. Yan, H., A. Bombarely, and S. Li, *DeepTE: a computational method for de novo classification of transposons with convolutional neural network.* Bioinformatics, 2020. **36**(15): p. 4269-4275.
22. da Cruz, M.H.P., et al., *TERL: classification of transposable elements by convolutional neural networks.* Briefings in bioinformatics, 2021. **22**(3): p. bbaa185.
23. Turnbull, R., et al., *Terrier: a deep learning repeat classifier.* Brief Bioinform, 2025. **26**(4).
24. Su, W., X. Gu, and T. Peterson, *TIR-Learner, a new ensemble method for TIR transposable element annotation, provides evidence for abundant new transposable elements in the maize genome.* Molecular plant, 2019. **12**(3): p. 447-460.

 doi:10.20944/preprints202511.1598.v1

25. Schietgat, L., et al., *A machine learning based framework to identify and classify long terminal repeat retrotransposons.* PLoS computational biology, 2018. **14**(4): p. e1006097.
26. Orozco-Arias, S., et al., *Inpactor2: a software based on deep learning to identify and classify LTR-retrotransposons in plant genomes.* Brief Bioinform, 2023. **24**(1).
27. Vaswani, A., et al., *Attention is all you need.* Advances in neural information processing systems, 2017. **30**.
28. Wolf, T., et al. *Transformers: State-of-the-art natural language processing*. in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 2020.
29. Ji, Y., et al., *DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome.* Bioinformatics, 2021. **37**(15): p. 2112-2120.
30. Devlin, J., et al. *Bert: Pre-training of deep bidirectional transformers for language understanding*. in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019.
31. He, S., et al., *Nucleic transformer: Deep learning on nucleic acids with self-attention and convolutions.* bioRxiv, 2021: p. 2021.01. 28.428629.
32. Dalla-Torre, H., et al., *Nucleotide Transformer: building and evaluating robust foundation models for human genomics.* Nat Methods, 2025. **22**(2): p. 287-297.
33. Wang, S., et al., *Linformer: Self-attention with linear complexity (2020).* arXiv preprint arXiv:2006.04768, 2006.
34. Beltagy, I., M.E. Peters, and A. Cohan, *Longformer: The long-document transformer.* arXiv preprint arXiv:2004.05150, 2020.
35. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning Representations by Back-Propagating Errors.* Nature, 1986. **323**(6088): p. 533-536.
36. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural Comput, 1997. **9**(8): p. 1735-80.
37. Sutskever, I., O. Vinyals, and Q.V. Le, *Sequence to sequence learning with neural networks.* Advances in neural information processing systems, 2014. **27**.
38. Luong, M.-T., H. Pham, and C.D. Manning, *Effective approaches to attention-based neural machine translation.* arXiv preprint arXiv:1508.04025, 2015.
39. Shorten, C. and T.M. Khoshgoftaar, *A survey on Image Data Augmentation for Deep Learning.* Journal of Big Data, 2019. **6**(1).
40. 40. Shaw, P., J. Uszkoreit, and A. Vaswani, *Self-attention with relative position representations.* arXiv preprint arXiv:1803.02155, 2018.
41. Suzuki, S., et al., *Genomic language models with k-mer tokenization strategies for plant genome annotation and regulatory element strength prediction.* Plant Mol Biol, 2025. **115**(4): p. 100.
42. Goodfellow, I., Y. Bengio, and A. Courville, *Deep learning*. Adaptive computation and machine learning. 2016, Cambridge, Massachusetts: The MIT Press. xxii, 775 pages.
43. Harris, C.R., et al., *Array programming with NumPy.* Nature, 2020. **585**(7825): p. 357-362.
44. Pedregosa, F., et al., *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 2011. **12**: p. 2825-2830.
45. Paszke, A., et al., *Pytorch: An imperative style, high-performance deep learning library.* Advances in neural information processing systems, 2019. **32**.
46. Abadi, M., et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems.* arXiv preprint arXiv:1603.04467, 2016.
47. Flynn, J.M., et al., *RepeatModeler2 for automated genomic discovery of transposable element families.* Proc Natl Acad Sci U S A, 2020. **117**(17): p. 9451-9457.
48. Dong, Y., J.-B. Cordonnier, and A. Loukas. *Attention is not all you need: Pure attention loses rank doubly exponentially with depth*. in *International conference on machine learning*. 2021. PMLR.