Briefings in Bioinformatics, 2023, **24(1)**, 1–10
https://doi.org/10.1093/bib/bbac511
Problem Solving Protocol

Inpactor2: a software based on deep learning to identify and classify LTR-retrotransposons in plant genomes

Simon Orozco-Arias, Luis Humberto Lopez-Murillo, Mariana S. Candamil-Cortés, Maradey Arias, Paula A. Jaimes, Alexandre Rossi Paschoal, Reinel Tabares-Soto, Gustavo Isaza and Romain Guyot

Corresponding authors. Simon Orozco-Arias, Computer Science Department, Universidad Autónoma de Manizales, Antigua Estación del Ferrocarrill, Manizalez, Colombia. Tel.: +57(606)8727272 - 8727709 Ext 102; E-mail: simon.orozco.arias@gmail.com; Alexandre Rossi Paschoal, Department of Computer Science, Bioinformatics and Pattern Recognition Group, Graduation Program in Bioinformatics, Federal University of Technology - Paraná, UTFPR, Cornélio Procópio, Paraná, 86300-000, Brazil. Tel.: +433133-3790; E-mail: paschoal@utfpr.edu.br; Gustavo Isaza, Systems and Informatics Department, Center for Technology Development - Bioprocess and Agro-industry Plant, Universidad de Caldas, St 65 #26-10, Manizales, Colombia. Tel.: +57(606)8781500 ext 13146; E-mail: gustavo.isaza@ucaldas.edu.co, Romain Guyot, IRD, 911 Av. Agropolis, 34394 Montpellier, France. Tel.: +334674160000; E-mail: romain.guyot@ird.fr

Abstract

LTR-retrotransposons are the most abundant repeat sequences in plant genomes and play an important role in evolution and biodiversity. Their characterization is of great importance to understand their dynamics. However, the identification and classification of these elements remains a challenge today. Moreover, current software can be relatively slow (from hours to days), sometimes involve a lot of manual work and do not reach satisfactory levels in terms of precision and sensitivity. Here we present Inpactor2, an accurate and fast application that creates LTR-retrotransposon reference libraries in a very short time. Inpactor2 takes an assembled genome as input and follows a hybrid approach (deep learning and structure-based) to detect elements, filter partial sequences and finally classify intact sequences into superfamilies and, as very few tools do, into lineages. This tool takes advantage of multi-core and GPU architectures to decrease execution times. Using the rice genome, Inpactor2 showed a run time of 5 minutes (faster than other tools) and has the best accuracy and F1-Score of the tools tested here, also having the second best accuracy and specificity only surpassed by EDTA, but achieving 28% higher sensitivity. For large genomes, Inpactor2 is up to seven times faster than other available bioinformatics tools.

Keywords: Inpactor2, LTR-retrotransposons, plant genomes, deep learning, neural networks, detection, classification

Simon Orozco-Arias received the B.S. degree in System and Computing Engineering and the Ph.D. Degree in Engineering (Bioinformatics) from the Universidad de Caldas, Colombia, in 2016 and 2022, respectively. He is currently working as professor and researcher with the Universidad Autónoma de Manizales, developing projects on bioinformatics, supercomputing, and data analysis. He has contributed to the solution of biological problems, especially, related to plant DNA. His current research interest includes the application of machine learning techniques to automatically extract knowledge from DNA data.
Luis Humberto Lopez-Murillo was graduated in chemical engineering in 2018 and candidate to M.Sc chemical engineering from Universidad Nacional de Colombia. Scientific researcher at Universidad Autónoma de Manizales. Research fields: (1) Deep learning applied to bioinformatics and (2) Process system engineering applied to membrane technologies. Member of research groups: (1) Ingeniería de software at Universidad Autónoma de Manizales (2) Grupo de Investigación en Aplicación de Nuevas Tecnologías (GIANT) at Universidad Nacional de Colombia.
Mariana S. Candamil-Cortés is a Biomedical Engineer from Universidad Autónoma de Manizales. Her research interests are bioinformatics, computational biology and data science.
Maradey Arias is a biomedical engineer from the Universidad Autónoma de Manizales (Colombia), and during her association with the Bioinformatics and Artificial Intelligence research group, she worked on the identification and classification of transposable elements in plants, using bioinformatics tools and machine learning techniques.
Paula A. Jaimes is a biomedical engineer from the Universidad Autónoma de Manizales, Colombia, belonging to the Bioinformatics and Artificial Intelligence research group since 2019. She has been working in genomics field, focused on the identification and classification of transposable elements in plants. Her current research interest includes the understanding of the DNA data through bioinformatic tools and machine learning techniques.
Alexandre Rossi Paschoal is an associate professor at the Dept of Computer Science - Federal University of Technology – Parana, UTFPR, Brazil. He received his Ph.D. in Bioinformatics at the University of São Paulo in 2012. His research includes data science, bioinformatics, and pattern recognition approaches to model and understand biological data.
Reinel Tabares-Soto received the B.S. degree in electronic engineer from the Universidad Nacional de Colombia, in 2009, the B.S. degree in systems and computer engineer from the Universidad de Caldas, Colombia, in 2016, the M.S. degree in engineering from the Universidad Nacional de Colombia, in 2017, and the Ph.D. degree in computer science from the Universidad Autónoma de Manizales, Colombia, in 2021. He has been a Professor and a Researcher with the Department of Electronics and Automation, Universidad Autónoma de Manizales, since 2014. He has been also a data science consultant for different public and private companies in Colombia since 2017. He is currently holding a Postdoctoral position at the Adolfo Ibañez University in the framework of the ethical and responsible algorithms project. His main research interests include steganalysis, machine learning, deep learning, bioinformatics, and high-performance computing.
Gustavo Isaza is a Ph.D. in Software Engineer from Universidad Pontificia de Salamanca, Spain, 2010, received a B.S. degree in system and computing engineering from Universidad Autónoma de Manizales, Colombia, in 1997. He obtained a postgraduate degree in networking software development from Universidad de los Andes, Colombia, in 1998. He is full Professor at Universidad de Caldas, Senior Researcher from MinCiencias (Ministry of Science, Technology and Innovation) of Colombia and a member of the GITIR Research Group and CDT Bioprocess and Agro-industry Plant researcher. He has published over 70 papers and conferences related to machine learning in cybersecurity, bioinformatics, AI in videogames, and distributed computing.
Romain Guyot is a research director at the Institut de Recherche pour le Développement. He has extensive experience in transposable elements genomics and bioinformatics.
Received: August 1, 2022. **Revised:** October 13, 2022. **Accepted:** October 26, 2022

Introduction

Transposable elements (TEs) are repeated sequences scattered throughout the genome [1]. They have the ability to move from one position in the genome to another, increasing their copy number [2]. The presence of these repeated elements in the genomes is now recognized as a powerful driver of evolution and biodiversity [3], domestication [3, 4] and variations in genome size [5]. According to their transposition mechanism, they are first divided into two classes: Class I or 'retrotransposons' and Class II or 'transposons'. They are further sub-classified hierarchically into subclasses, orders, superfamilies, lineages and families [6, 7]. Long terminal repeat (LTR) retrotransposon (or LTR-RT), an order of Class I elements [8] comprise the most abundant elements in plant genomes [9, 10]. LTR-RTs move via a copy-and-paste mechanism, using an RNA intermediate that is reverse transcribed into cDNA and introduced into the genome by a integrase coded by the full LTR-RT copies [6, 11]. They are sub-classified into two superfamilies found in a large number of eukaryotes, namely Ty1/*Copia* and Ty3/*Gypsy* [12], which differ structurally by the order of the internal coding domains [13]. They can be further sub-classified into lineages according to similarities of their domains [14].

Machine Learning (ML) techniques have been used to solve several genomic and evolutionary problems of biological systems [15]. ML is defined as the use of calibrated computational algorithms based on previous experiences through statistical inference on the data to make predictions in classification and regression problems [16, 17]. Its main function is to tune the parameters needed to optimize performance on training data and subsequent input data [18, 19].

Some researches have used ML models for TE identification, for example, Orozco-Arias *et al.* (2019) [13] reviewed the use of ML for the analysis of TEs and Loureiro *et al.* (2013) [20], Nakano *et al.* (2018) [21] and Panta *et al.* (2021) [22] studied how to improve the accuracy and performance of TE classification. Recently, preprocessing techniques and coding schemes that enable deep classification of TEs have been studied [23, 24]. Besides, software such as RED [25], PASTEC [26], TEClass [27] and TransposonUltimate [28] apply ML techniques such as Support Vector Machines, Random Forests, Hidden Markov Models, K-nearest Neighbors, Neural Networks (NN) and graphical models due to feature extraction, process automation and faster algorithm executions.

To identify and classify TEs, bioinformaticians have developed many tools, techniques and methods, including structure-based, homology-based, *de novo* and comparative genomics [29–31]. The most accurate tools combine several methods to improve their results, such as LTR-FINDER [32], EDTA [33] and Inpactor version 1 [34]. Nevertheless, this makes the tools slower in execution especially in large genomes, taking hours or even days, which, together with high variability and redundancy of TEs [35], makes it unfeasible to process the large amount of genomic data that are released every day [36].

In recent years, several datasets consisting of thousands of TEs from various species have been created and published, such as Repbase [37], RepetDB [38], PGSB Plants DB [39] and InpactorDB [40]. These datasets constitute valuable resources for improving tasks such as TE detection and classification and have motivated the proposal and evaluation of ML techniques to obtain substantial results in terms of accuracy and speed in executing these tasks [20, 31].

Although good results are obtained using current ML techniques, such as ordinary NN, recent advances have shown that Deep Neural Networks (DNNs) can achieve better results, in which non-parametric models based on NN are implemented to adapt associations between input and output data [41, 42]. In this field, several DNN architectures have been published, such as the Fully Connected Neural Network (FNN) by [21], the Convolutional Neural Network (CNN) with 2D representation of sequences by [42, 43] and the 1D CNN for classifying TEs into superfamilies by [44]. However, none of these NNs have been integrated into a single software that automatically detects and classifies TEs in relatively short times.

Additionally none of the DL approaches or current tools (with the exception of TEsorter [45]) perform the task of classifying LTR-RTs at the lineage/family level to obtain accurate results. Here, we present Inpactor2, a novel method and tool based on four NN that detect and classify LTR-retrotransposons automatically. This tool was executed in plant genomes up to 2.2 Gb and obtains results up to seven times faster than state-of-the-art tools decreasing the execution time from more than 20 hours to less than three. Moreover, using the rice genome *Oryza sativa*, Inpactor2 gets the best performance in accuracy (96.1%) and F1-Score (91.9%) and the second best in specificity (97.7%), precision (92.7%) and in false discovery rate (FDR) (7%) of the tools tested here. Inpactor2 is freely available at https://github.com/simonorozcoarias/Inpactor2. Additional information about NNs architectures used, and their hyper-parameters can be found at https://github.com/simonorozcoarias/Inpactor2/tree/main/NN_architectures.

Materials and methods

Genomic datasets used for training of NNs

Inpactor2 was designed to perform three main tasks in the analysis of LTR-RTs, detection, filtering and classification. Each task is done by a different NN. Thus, detection is done by a CNN named Inpactor2_Detect, filtering by an FNN called Inpactor2_Filter and classification by another FNN named Inpactor2_Class. Thus, each architecture needed different datasets to be trained. For example, Inpactor2_Detect was trained on a dataset consisting of 70 000 sequences of 50 000 bases in length. To create the target sequences (those regions with LTR-RTs in them), we randomly obtained an LTR-retrotransposon from InpactorDB [40] and placed it at a random position. Then, the remaining one was filled with sequences corresponding to other genomic features obtained from [24] (DOI 10.5281/zenodo.4543904). For negative sequences (those without LTR-RTs within), sequences of genomic features other than LTR-retrotransposons were again obtained from [24] (DOI: 10.5281/zenodo.4543904). This dataset was composed of 35 000 sequences with LTR-RTs (target sequences) and 35 000 sequences without these elements (negative sequences).

On the other hand, Inpactor2_Filter was trained using a dataset with binary labels, where zero corresponded to intact sequences and one corresponded to non-intact. All the sequences presented in InpactorDB (67 241 elements) were used as intact elements, while non-intact sequences were taken from those filtered by [40] and corresponded to (a) sequences with a LTR-RT inserted into another (from different superfamilies or lineages), (b) sequences of lengths shorter or larger than those reported in the Gypsy Database [46] (with a tolerance of 20%) and (c) sequences with TEs class II inserted into LTR-RTs. In total, this dataset contained 105 657 sequences. As features, Inpactor2_Filter used k-mer frequencies using $1 \leq k \leq 6$.

Finally, Inpactor2_Class was trained using InpactorDB after k-mer frequencies extraction in the same way as Inpactor2_Filter. This dataset comprised 67 241 LTR-RTs classified at the lineage/

Table 1. Plant genomes used in the execution time tests

Species	Assembly size (Mb)	Accession number
Arabidopsis thaliana	121.2	GCF_000001735.4
Oryza sativa	379.1	GCF_001433935.1
Coffea canephora	579.4	GCA_900059795
Solanum lycopersicum	839.1	GCF_000188115.4
Coffea arabica	1126.4	GCF_003713225.1
Zea mays	2252.8	GCF_902167145.1

family levels. *k*-mer frequencies were estimated in each sequence by calculating all possible *k*-mers with a maximum length of six nucleotides and later counting the number of occurrences [23, 24].

Benchmarking the performance of Inpactor2

The reference *O. sativa* genome [47–51], was selected for testing the software due to its high-quality assembly, small genome size (389 Mb) and quality of its genes and TEs annotations. The *O. sativa* genome was identical to the one used by [33] to compare the results with benchmarking tools in this study. This work used the standard library v6.9.5 created by [33] based on the *O. sativa* L. ssp. *japonica* cv. 'Nipponbare' v. MSU7 genome and RepeatMasker v4.0.8 [52] with the following parameters '-pa 36 -q -no_is -norna -nolow -div 40 -cutoff 225'.

Additionally, six different plant genomes (Table 1) were used to test the execution times of Inpactor2 by assessing different genome sizes and TE compositions. The genomes were downloaded from NCBI and analyzed with Inpactor2 using the following parameters (-m 15000 -n 1000, -i no, -d no, -C 1, -c yes -a no), as suggested in [53]. Finally, EDTA was run with the same genomes to compare its execution times with Inpactor2. EDTA was executed using EDTA_raw.py script, –type ltr, and the other parameters by default.

Libraries of LTR-RTs of the species shown in Table 1 were then created using Inpactor2 (with and without filtering with the -c flag) and EDTA. In addition, two species that were not contained in the training data were used, such as *Coffea humblotiana* [54] and *Gardenia jasminoides* [55]. These libraries were then annotated using repeatMasker and compared with the proportion of genomes corresponding to LTR-RTs according to the papers where the genomes were reported. A workstation with AMD Ryzen Threadripper 3970X 32-Core Processor, 128 Gb in RAM memory and a GPU Nvidia RTX 2080 super was used to perform all the experiments.

To evaluate the performance of Inpactor2 compared with other software, a similar methodology to the one proposed in [33] was followed. First, Inpactor v.1.0 [34], TEsorter v.1.3 [45], Transposon Ultimate v.1.0 [28], LTR_retriever v.2.9 [56] and LTRharvest [57] were selected for benchmarking given their methodologies for classifying LTR-RTs to the superfamily level. A workflow was established for each software, initially using LTR_FINDER v.1.0.7 as the LTR-RTs detector. Then, the *O. sativa* genome was annotated with RepeatMasker and performance metrics were extracted for each workflow. The metrics evaluated were: accuracy, precision, specificity, sensitivity, FDR and F1-score. Figure 1 shows the schematic representation of the benchmarking metrics. In this study, TP, FN, TN and FP are the number of nucleotides belonging to each category (Figure 2).

The script called 'lib-test.pl', included in the EDTA toolkit [33], was used to extract the six metrics. Since this study only focused on the LTR-RT category, so the script was executed using the -cat ltr parameter to perform the comparative evaluation.

Results

General architecture of Inpactor2

Inpactor2 is composed of four NNs, each one performs one function in the pipeline. First, Inpactor2 receives a genome assembly in FASTA format, then each sequence in the input file is cut into 50 kb sections, without overlapping. Each section is converted into a 2D-representation using one-hot encoding. This coding scheme generates a 5x*n* matrix of *ones* and *zeros*, where each row corresponds to the possible nucleotides (A, C, T, G or N) and the columns represent the bases present in the sequence of length *n*. This coding puts a 1 in the position of the row that corresponds to the base of the sequence. For an example see [43] and the section 'Inpactor2_*K*-mers network definition'. Next, a CNN called Inpactor2_Detect is used to predict which section contains LTR-RTs and these segments are retained for further analysis. This network has the only function of retaining those sections of interest in the genome and eliminating those that do not contain LTR-RTs (according to the predictions). In this way, the execution time and memory required for the following steps is optimized. Next, LTR_FINDER is run on the sections that were predicted to contain LTR-RTs inside by Inpactor2_Detect to search for the start and end positions of the detected LTR-RTs. This step is executed in parallel to reduce the execution time. After, a CNN called 'Inpactor2_*K*-mers' is used to count *k*-mer frequencies in the extracted LTR-retrotransposons. This CNN is intended to extract features required by the following NNs of the pipeline in a time-efficient way (See 'Inpactor2 *K*-mers network definition' section). Intact and potentially complete LTR-retrotransposons are filtered and retained based on the *k*-mer frequencies and a FNN called 'Inpactor2_Filter'. Finally, a FNN named 'Inpactor2_Class' is used to classify the elements into lineages. Figure 3 shows a schematic of the general structure of Inpactor2.

Inpactor2 can be executed using several cycles (from one to five) of analysis, where each cycle splits differently the input sequences in order to predict the elements that remain split in any of the partitions. This behavior is controlled with the parameter -C (upper case), and its default value is 1. Additionally, Inpactor2 can be performed using different structural parameters to filter LTR-RTs, such as minimum and maximum LTR-RT length, LTR domains starting with TG and finishing with CA, and target site duplication (TSD) before and after the element. These parameters are given with the flags -M, -m, -i and -d, where their default values are 2000, 28000, yes and yes, respectively.

After finishing all cycles, Inpactor2 removes the predictions that are non-maximal, following the same methodology of the Yolo architecture [58]. An analogous concept is adapted from there, namely the intersection over union (IOU) operation (also named as Jaccard index). Here, the IOU operation between two LTR-RT predictions is defined as the number of nucleotides overlapping in both predictions (intersection) over the total number of nucleotides covered in the chromosome by both predictions (union) (Figure 4). If two predictions have an IOU score > 0.6, then, Inpactor2 only keeps the one with best prediction score. This score is calculated as the average of the probabilities obtained by each NN (Inpactor2_Detect, Inpactor2_Filter and Inpactor2_Class). These probabilities can be consulted in the Inpactor2's output file named 'Inpactor2_predictions.tab'.

The IOU score is defined in the Equation 1

$$IOU = \frac{max(0, min(Y1, X1) - max(Y0, X0))}{max(Y1, X1) - min(Y0, X0)}, \tag{1}$$

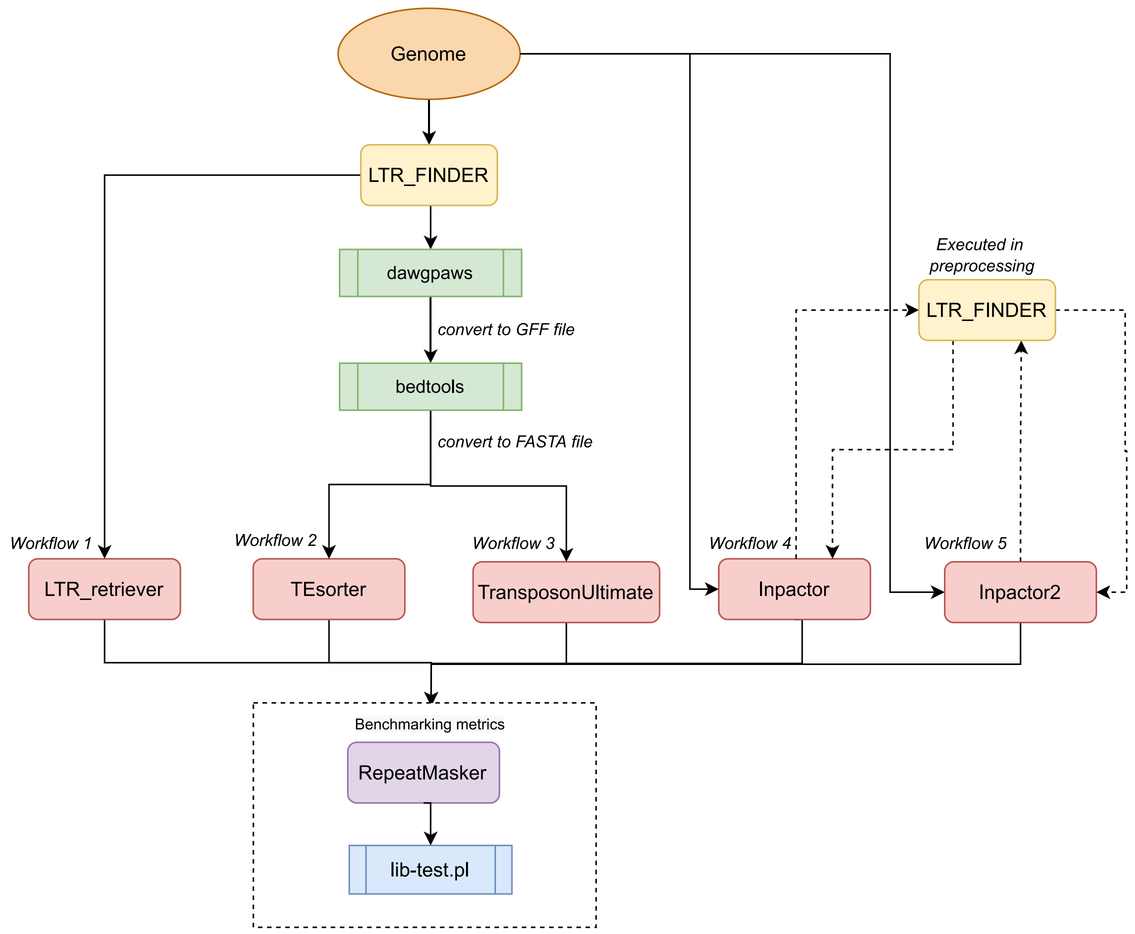

Figure 1. Workflow of the benchmarking process. Firstly, LTR_FINDER was executed as detector software for each workflow. Then, because TEsorter and TransposonUltimate use as input a FASTA-format file, dragpaws v.3.0 (http://dawgpaws.sourceforge.net/man.html) was executed to convert the LTR_FINDER's output to GFF and bedtools v.2.29.2 (https://bedtools.readthedocs.io/en/latest/index.html) was utilized to convert it to FASTA. Both, Inpactor version 1 and Inpactor2 used LTR_FINDER in their own workflow. LTR_Retriever was the unique tool that uses the raw LTR_FINDER's output. Finally, using the library created by each workflow, RepeatMasker was executed and the perl script named lib-test.pl [33] was utilized to get the performance metrics.

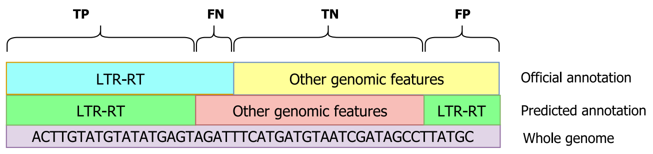

Figure 2. Metrics evaluated for each LTR-RT classifier. Based on [33]. TP corresponds to true positive, FN to false negative, TN to true negative and FP to false positive.

where X0 and Y0 are the beginning positions of predictions 1 and 2, and X1 and Y1 are the ending positions of predictions 1 and 2, respectively (Figure 4).

Inpactor2_Detect network definition

To reduce the execution time, Inpactor2 first splits the input sequences into 50 kb sections and predicts which section may contain LTR-retrotransposons. To do this, a CNN was designed based on the TERL architecture [43]. Inpactor2_Detect is composed of three convolutional layers, each one followed by a max pooling layer. After the convolutional layers, two fully connected layers are used to obtain the predictions of 1000 and 500 neurons, respectively (Figure S2). The activation function in all layers was ReLu, whereas SGD was used as optimizer and binary cross-entropy was used as loss function. The network was trained using 100 epochs and a batch size of 64. The Inpactor2_Detect performance metrics can be consulted in Table S1.

Inpactor2_*K*-mers network definition

The software proposed herein performs pre-processing of the information stored in the extracted LTR-RT sequences. This algorithmic treatment includes the computation of *k*-mer frequencies, a principal component analysis and scaling. Although there are bioinformatic algorithms to compute the *k*-mer frequencies, a CNN is proposed in this paper for counting 1-mers to 6-mers in DNA sequences. However, to accomplish this task, the DNA sequence must be first transformed to a digital encoding. According to [43], a suitable representation of the DNA is the one-hot encoding, so a modification of this is used henceforth. The one-hot encoding used here only has five rows: A (adenine), C (cytosine), G (guanine), T (thymine) and N (unidentified nucleotide).

To understand how a CNN can be used to compute *k*-mer frequencies, two examples are considered in Figure 5. The first one is computing the amount of 'A' in a DNA sequence 'ACTGCCTAA', whereas the second example is computing the amount of 'CT' in the same DNA sequence. According to the Figure 5, the weights and biases can be set manually to compute the frequency of any *k*-mer. For example, the amount of a specific *k*-mer in a DNA

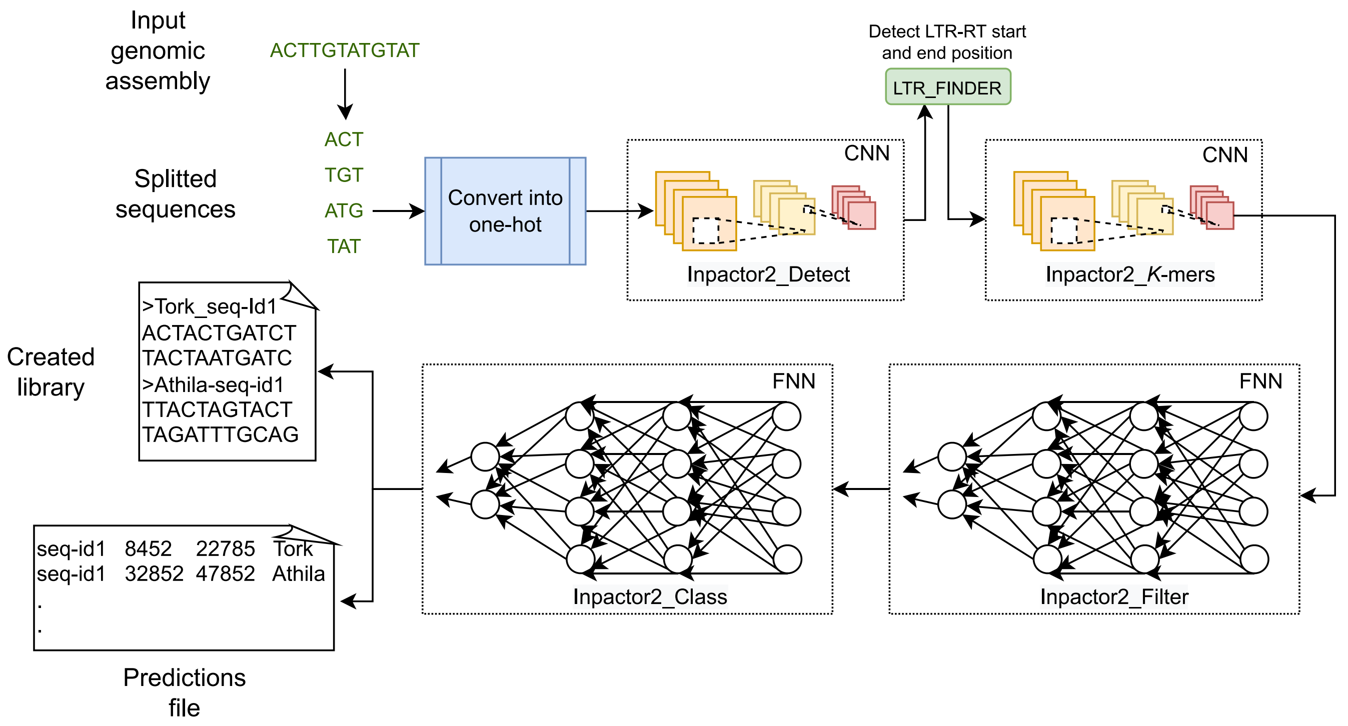

Figure 3. General schematic representation of the Inpactor2 workflow. The tool's core is composed by four NN. Inpactor2 receives as input the genomic assembly, then it splits the sequence into non-overlapping sections of 50 kb length. As outputs, Inpactor2 creates a library in FASTA format with the detected and classified LTR-RTs and a tabular file with the predictions made by the three NNs. A detailed graphical schema of the Inpactor2 workflow and each section can be found at Figure S1.

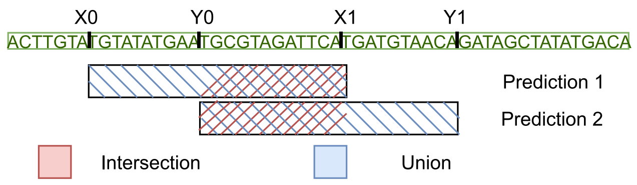

Figure 4. IOU operation between two overlapping LTR-RT predictions along a chromosome sequence.

sequence can be computed if the weight matrix is set equal to the 2D representation of that k-mer and the bias is set to $1 - k$.

Considering that a filter can have more than one dimension, these can be used to compute all the k-mer frequencies at once, accelerating the counting time compared with the conventional counting method (Table S2). The final CNN architecture is illustrated in Figure S3.

Inpactor2_*K*-mers takes mathematical representations of DNA sequences with dimensions of 5x50000x1 and extracts the frequencies of 1–6-mers in lexicographic order. Therefore, the frequencies of $4^1+4^2+4^3+4^4+4^5+4^6 = 5460$ k-mers can be computed by the developed CNN.

Inpactor2_Filter network definition

To obtain intact LTR-retrotransposons, ML-based experiments were developed. For this, sequences from InpactorDB [40] named class 0 (intact elements) are used, whereas the elements that were removed in each of the filters proposed in the same study were taken as non-intact sequences named class 1.

Using the produced dataset, a NN based on the FNN proposed in [21] was trained and tuned. The developed architecture, called Inpactor2_Filter, is shown in Figure S4. For each of the layers, a dropout of 0.5 and a ReLu activation function were utilized using BatchNormalization with a momentum of 0.99. For the prediction layer, a softmax activation function was used. Adam optimization algorithm was applied to find a suitable configuration of the NN and categorical Cross-entropy was used as the loss function. Furthermore, regularization l1 was added to kernel with value of 0.0001 and l2 to bias with a value of 0.01. These regularizers were applied to the three hidden layers. For the training stage, 200 epochs were used and the batch size was set to 128.

Performance metrics are shown in Table S1 to evaluate the correct generalization of the model.

Inpactor2_Class network definition

Following detection and filtration, LTR-RTs were classified into lineages. A FNN was designed with three hidden layers, each one with 200 neurons. The Inpactor2_Class structure is similar to Inpactor2_Filter (Figure S4), but with 13 output neurons to implement the multi-class classification. This network was trained for 200 epochs with 128 as batch size, using InpactorDB [40] sequences converted to k-mer frequencies. Inpactor2_Class obtained a performance of 0.98 in accuracy, precision, recall and F1 score (Table S1).

Inpactor2_utils

In addition to the main component of Inpactor2, an extra script that contains utilities for the LTR-RT analysis was released, such as deletions of characters different from nucleotides (A, C, T, G or N), calculation of k-mer frequencies with $1 \leq k \leq 6$ using Inpactor2_*K*-mers architecture, re-training Inpactor2_Class to specialize the NN for a specific group of species, among others. Inpactor2_utils is available in the same repository than the main script.

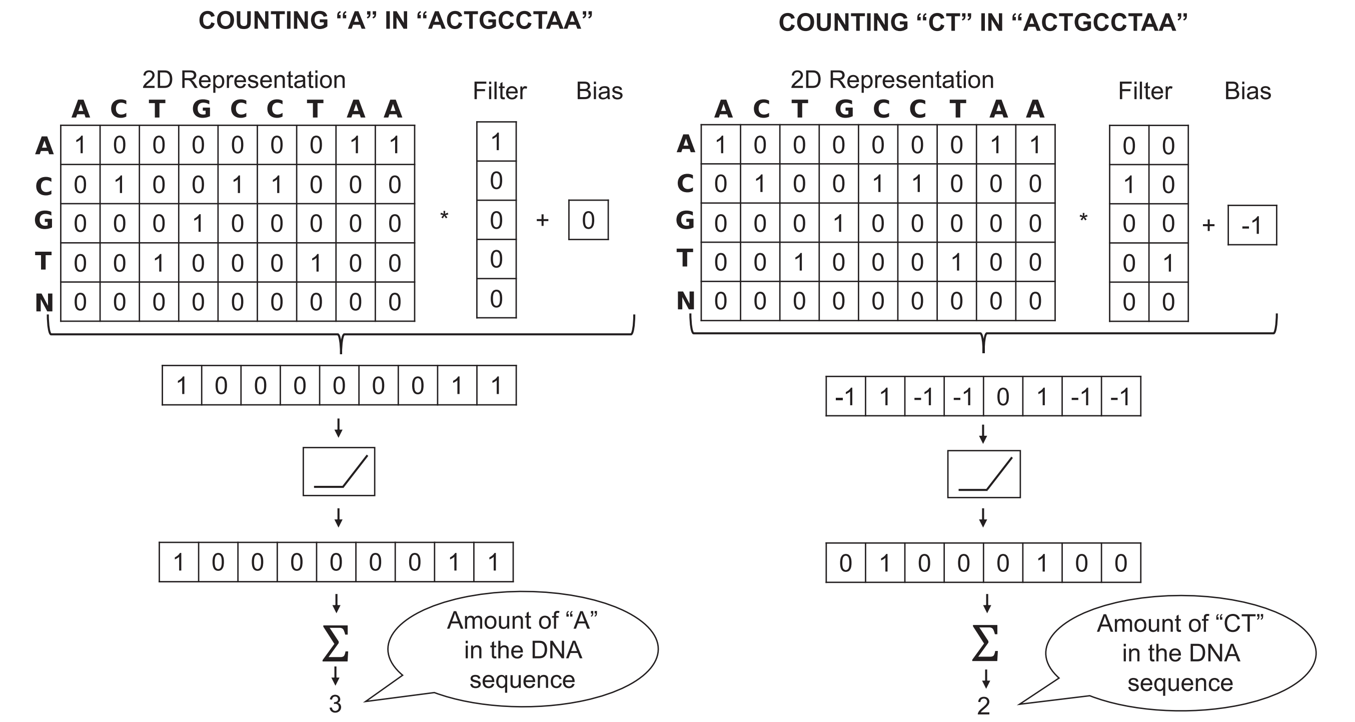
COUNTING "A" IN "ACTGCCTAA"
2D Representation
Filter
Bias
A C T G C C T A A
Amount of "A" in the DNA sequence
3
COUNTING "CT" IN "ACTGCCTAA"
Amount of "CT" in the DNA sequence
2

Figure 5. Convolutions used for *k*-mer frequencies computations.

Benchmarking results

Following the proposed methodology, an evaluation of sensitivity, accuracy, precision, specificity, FDR and F1-score metrics, as well as false positive, false negative, true positive and true negative rates was performed for the selected software (Figure 6 and Table S3). An accuracy of 96.1% was obtained for Inpactor2, representing the highest value for that metric, as well as an F1-score of 91.9%. However, for specificity, precision and FDR, Inpactor2 obtained the second best values with 97.7%, 92.7% and 7%, respectively only surpassed by EDTA. Nevertheless, EDTA showed the lowest value of sensitivity, whereas Inpactor2 obtained 28% more in this metric.

Execution time comparisons

For each workflow, the execution time was counted to determine which of the selected software was the fastest for the detection and classification of complete LTR-RTs (Figure S5). The best execution time is given by Inpactor2, with 5.47 minutes (in the *O. sativa* genome), constituting the fastest software for the creation of complete LTR-RT libraries.

Additional execution time tests were performed using different genomes sizes ranging from 121.2 Mb for *Arabidopsis thaliana* to 2,252.8 Mb for *Zea mays*. In these tests, only EDTA and Inpactor2 were executed. Table S4 shows that Inpactor2 is faster than EDTA, specially with larger genomes (i. e. up to seven times faster for the biggest genome tested, *Z. mays*).

The next step consisted in creating libraries of the genomes shown in Table 1 and of the species of *C. humblotiana* and *G. jasminoides* using three different flows, Inpactor2 doing the curation of the sequences (with the Inpactor2_Filter network), Inpactor2 without process of curating and EDTA. At the end, the libraries were used to annotate the LTR-RTs and compare the proportions of each genome that correspond to each library against what was reported in the original papers. Figure S6 shows a large variability between the different streams with respect to the original annotation. This could be because there is no standard annotation process for TEs and therefore different investigations use different characteristics to consider whether or not a sequence should be in the annotation library. When specifically comparing EDTA and Inpactor2 (Figure S7) with filtering turned on (at the library level) we found that some of the LTR-RTs contained in the Inpactor2 library were the representation (in average) of up to 41 models in the EDTA library (Figure S7I). This observation is in line with that reported by [59] in other organisms such as *Drosophila melanogaster*. However, it is interesting that although EDTA appears to create many more models than Inpactor2, at the annotation level (Figure S6) the difference is much less evident.

Discussion

TEs represent the main component of plant genomes. They are sources of mutations and polymorphisms responsible of the phenotype variability in numerous crop species. For instance, they influence grape color [60], pigment instability in corn kernels (from completely yellow to spotted kernels to completely purple), tomato shape and size [61], potato skin color [62], orange color and flavor [63] and the presence or absence of fuzzless in peach skin [64]. In addition to phenotypic implications, others researches have shown that these elements have key roles in other aspects such as contributing to intra-species diversity [65], influencing gene expression [66, 67] and potentially adaptation to climate changes [68]. With the improvement of genomic sequencing techniques, a large amount of plant genomes is now available in public sequence databases. In addition the sequencing of large and complex genomes, containing a high amount of TEs, is becoming possible at moderate costs. These genomes make possible the analysis of TEs with the accessibility of a variety of annotation pipelines using different strategies, such as structure-based, homology-based, *de novo* and comparative genomics-based [13, 20, 29–31] and, lately, ML-based (or alignment-free) [21, 22, 24, 28, 31, 40, 42–44].

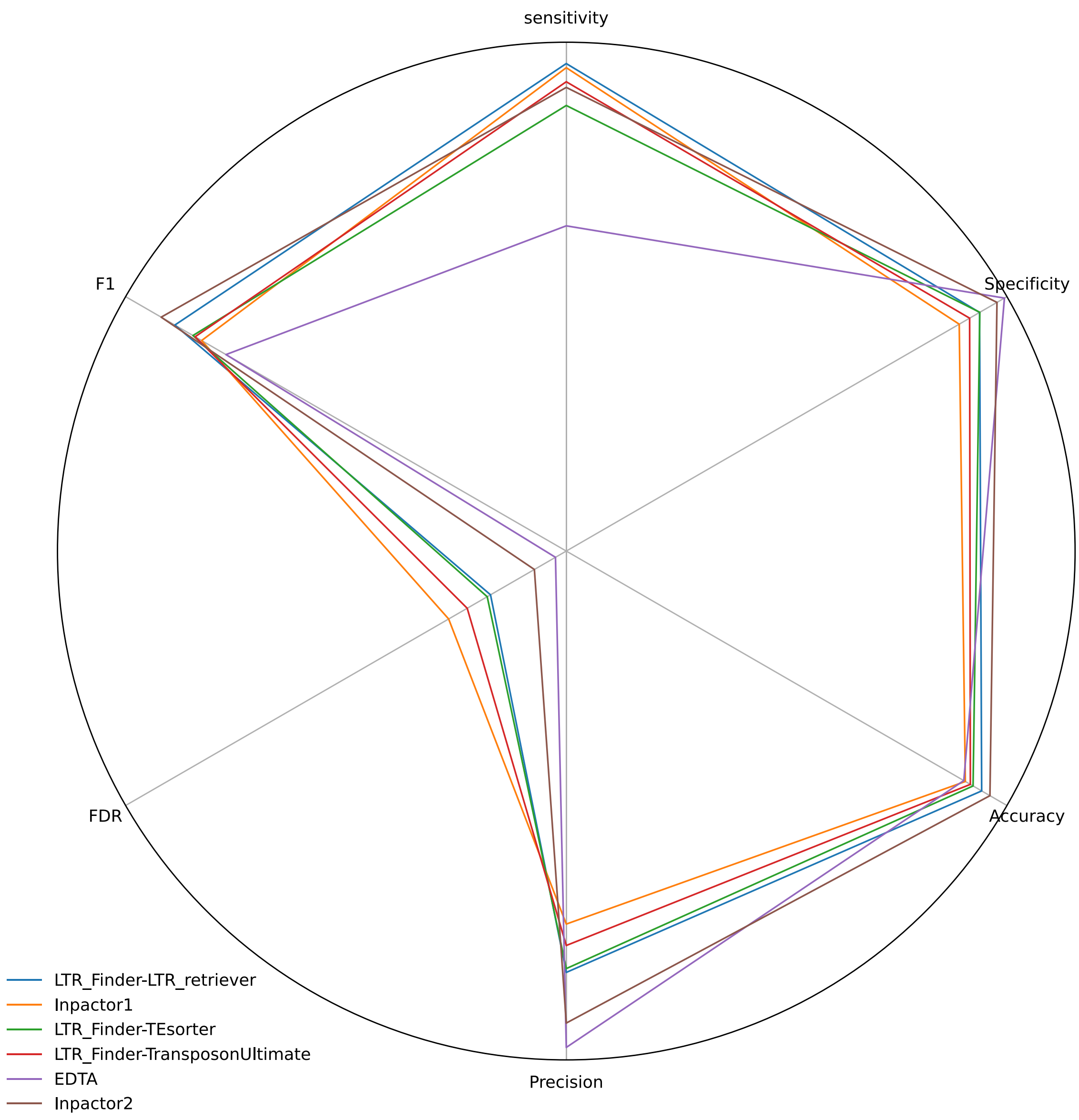

Figure 6. Benchmarking of LTR-RT classifier tools. All the results obtained by each tool in each of the metrics can be consulted in Table S3.

ML-based (or alignment-free) strategies are clearly well adapted for analyzing numerous and large genome size, but require a high-quality and representative training dataset to be applied with reliability on different species; otherwise, they will not make good predictions on species far from those contained in the training dataset. However, at the computational level, the analysis of genomic datasets (and more particularly TEs) is particularly complicated with ML-based approaches due to the size, the relatively unstructured format and the complex visualization of the data. Moreover, since each nucleotide is represented by a letter (categorical data type), a transformation (or a feature extraction) is required to process the data. *K*-mers have proven to be crucial features [24] to implement ML models [69, 70] and NN architectures [21, 44]. Nevertheless, counting *k*-mer frequencies requires computational time and displays issues with space consumption and scalability [71]. This fact can negatively affect the entire process of the software, specially in genomes with many LTR-RTs, becoming a bottle neck in NN-based tools. Thus, by taking advantage of GPUs, Inpactor2_K-mers offers the possibility to compute *k*-mer frequencies faster than using CPU-based and conventional methods (Table S2) overcoming this problem. By using filters from a convolutional network, each of the required 5460 *k*-mer frequencies can be calculated from matrix operations, which libraries such as Tensorflow can execute on CPU or GPU without user input. In addition, by simultaneously counting all *k*-mers and using the batch size parameter (depending on the amount of memory available on the GPU device) the computation time can be reduced considerably.

Inpactor2 was designed with the objective to produce highly accurate libraries of complete LTR-RT sequences in plant genomes and to optionally annotate them in genome assemblies. Inpactor2 is a hybrid tool that basically combines four NNs for detection, providing sensitivity and speed of execution, with a structure-based software (LTR_FINDER) to extract potential candidates and finally with a homology-based tool (RepeatMasker) to annotate all fragments in assemblies.

Further Results obtained by Inpactor2_Detect and LTR_FINDER at the full region level (Figure S8 and Table S5) using the genomes of *A. thaliana* and *O. sativa* allowed us to better understand how the hybrid approach processes the genomic data. We were able to conclude that only 1.3% in *A. thaliana* and 3.7% in *O. sativa* of all the regions into which the genome was divided contained elements according to LTR_FINDER, but were not detected by Inpactor2_Detect (column 'LTR_FINDER only'). On the other hand, we found that 45.8% in *A. thaliana* and 25% in *O. sativa* of all regions did not contain LTR-RTs according to both approaches (column 'Both negative'). This allows the software to not run LTR_FINDER (which is much slower than Inpactor2_Detect) only on these portions of regions that were found by Inpactor2_Detect.

The methodology implemented in Inpactor2 (in Inpactor2_Detect) allows increasing the sensitivity of detection compared with other software, such as EDTA and LTR_FINDER. The FDR remains very low due to the filtration performed by Inpactor2_Filter, which retains only intact or complete sequences. This curation process is a crucial task to achieve good quality masking and annotation because if element fragments (for example soloLTRs) are present in the library, which come from intact (or complete) LTR-RTs that are also in the library, the annotation could show an over-estimation of the element contribution. Since it would take into account both the whole element and its fragments [72]. Finally, Inpactor2_Class assigns a lineage to each predicted complete element. This deep level of classification is very rare in classification tools, which typically reach only the superfamily level (i.e. *Copia* an *Gypsy*), such as in EDTA, TransposonUltimate and LTR_retriever.

Inpactor2 is composed of four NNs, each one designed for a specific task and trained with a different and specially designed dataset. This approach allows each NN to have a relevant performance. For example, Inpactor2_Detect has an accuracy, precision, recall, and F1-score of 97%, Inpactor2_Filter has 91% in the same metrics and Inpactor2_Class has 98% in the metrics mentioned above (Table S1). Altogether, these NNs (along with Inpactor2_K-mers) integrate a pipeline that is up to seven times faster than popular tools such as EDTA (Table S4), while maintaining the performance showed by software such as LTR_FINDER, LTR_retriever and Inpactor version 1. Inpactor2 was designed to be easy to install (within an Anaconda environment) and to execute. Also, Inpactor2_utils.py provides more utilities to facilitate the implementation of the entire pipeline. Finally, Inpactor2 can be easily re-trained with data from specific plant genera or orders to create specialized models, thus increasing the performance of the tool and accelerating studies that simultaneously cover many species.

Together our data suggest that Inpactor2 can be a reliable and rapid tool to build libraries and annotate LTR-RTs in plant genomes and to analyze these sequences in a comprehensive and reproducible manner. As an a example, we searched within the *O. sativa* genome for five well-known LTR-RTs and all of them were found in the Inpactor2's library with a BLASTn identity higher than 90% (Table S6). However, it should be kept in mind that the performance of Inpactor2 depends on two essential factors: the quality and representativity of the dataset used for training Inpactor2 and of course the quality and the contiguity of the genome assemblies analyzed. In the near future, more large-scale LTR-RT analysis (such as the LTR-RT analysis of 300 plant species released by [73]) and high-quality genome assemblies will contribute to generate more datasets that can be used for re-training Inpactor2, increasing again its performance. Also as future work, it is proposed to generate and include datasets of other groups of organisms such as animals and fungi, as well as to include other TE taxa following the same methodology shown in InpactorDB.

Conclusion

Inpactor2 is a four-NN-based tool that reduce the execution time of complete LTR-retrotransposon library creation from plant genomes. This software detects, and classifies complete LTR-retrotransposons in few minutes (up to 26 minutes for *C. arabica*, 1.1 Gb), speeding up the computational time up to 7.1 times compared with EDTA (using *Z. mays* genome) and constituting the fastest tool tested in this study. Our benchmarking suggests that Inpactor2 is more sensitive, accurate and has a higher F1-Score than EDTA. Also, it has a higher specificity and precision and lower FDR than LTR_FINDER, Inpactor version 1, TEsorter and TransposonUltimate.

Key Points

- The hybrid approach used by Inpactor2 allows the creation of quality LTR-retrotransposon libraries, maintaining a high level of precision, accuracy and sensitivity and keeping a low number of false positives.
- Inpactor2 can be run using CPUs + GPUs, speeding up the execution time up to 7 times, being the fastest software in the creation of libraries of the tested software. This allows to analyze more genomes in less time, being useful for large scale analysis.
- Although other software is capable of identifying and classifying TEs in genomes, such as REPET, Inpactor2 is the first available tool that integrates the process of detection, curation and classification at the lineage level in a single software package and in a reasonable computation time. It is also easy to install and use, eliminating the need for manual operations.
- Inpactor2 is the first NN-based tool to detect LTR-retrotransposons *de novo*. It can be installed in an anaconda environment and can be run in a single Python command.

Data availability statement

The data underlying this article are available in the article and in its online supplementary material.

Supplementary Data

Supplementary data are available online at http://bib.oxfordjournals.org/.

Author contributions statement

R.G., G.I. conceived the experiment(s); S.O.-A., L.H.L.-M., M.S.C.-C., M.A. and P.A.J. conducted the experiment(s); S.O.-A. and R.T.-S. analyzed the results; S.O.-A., L.H.L.-M., M.S.C.-C., M.A., P.A.J., A.R.P., R.T.-S., R.G. and G.I. wrote and reviewed the manuscript.

Acknowledgements

We would like to thank the following HPC bioinformatics platforms for their support: the French Bioinformatics Institute (https://www.france-bioinformatique.fr), IRD (https://bioinfo.ird.fr/), Genotoul (http://bioinfo.genotoul.fr/), BiRD (https://pf-bird.univ-nantes.fr/) and Migale Bioinformatics facility (http://migale.jouy.inra.fr/). We also thank to Andrea Gonzalez for English editing.

Funding

S.Orozco.-A. is supported by a PhD grant from the Ministry of Science, Technology and Innovation (Minciencias) of Colombia, Grant Call 785/2017. The authors and publication fees were supported by Universidad Autónoma de Manizales, Manizales, Colombia under project 752-115 and Universidad de Caldas under project 0319120 and 0277920. This work was supported by Minciencias-Ecos Nord N° C21MA01 and 285-2021 and STICAMSUD 21-STIC-13 (Call STIC AmSud Latin America with France, Colombia, Chile, and Brazil from TELearning Project 2021-22). Also, this work was partly supported by ARP grant NAPI Bioinformática by Fundação Araucária (# 66/2021) and NVIDIA from the GPU Grant Program 2019 - Accelerated Data Science Call.

References

1. Mita P, Boeke JD. How retrotransposons shape genome regulation. *Curr Opin Genet Dev* 2016;**37**:90–100.
2. Keidar D, Doron C, Kashkush K. Genome-wide analysis of a recently active retrotransposon, au sine, in wheat: content, distribution within subgenomes and chromosomes, and gene associations. *Plant Cell Rep* 2018;**37**(2):193–208.
3. Lisch D. How important are transposons for plant evolution? *Nat Rev Genet* 2013;**14**(1):49–61.
4. Kawase M, Fukunaga K, Kato K. Diverse origins of waxy foxtail millet crops in East and Southeast Asia mediated by multiple transposable element insertions. *Mol Genet Genomics* 2005;**274**(2):131–40.
5. Ibarra-Laclette E, Lyons E, Hernández-Guzmán G, *et al.* Architecture and evolution of a minute plant genome. *Nature* 2013;**498**(7452):94–8.
6. Wicker T, Sabot F, Hua-Van A, *et al.* A unified classification system for eukaryotic transposable elements. *Nat Rev Genet* 2007;**8**: 973–82.
7. Orozco-Arias S, Isaza G, Guyot R, *et al.* A systematic review of the application of machine learning in the detection and classification of transposable elements. *Peer J* 2019;**7**:1–29.
8. Ramakrishnan M, Satish L, Sharma A, *et al.* Transposable elements in plants: Recent advancements, tools and prospects. *Plant Mol Biol Rep* 2022;**40**:1–18.
9. Bennetzen JL, Wang H. The contributions of transposable elements to the structure, function, and evolution of plant genomes. *Annu Rev Plant Biol*, **65**(1):505–30, apr 2014.
10. Grandbastien MA. LTR retrotransposons, handy hitchhikers of plant regulation and stress response. *Biochim. Biophys. Acta, Gene Regul. Mech.* 2015;**1849**(4):403–16.
11. Boeke JD, Garfinkel DJ, Styles CA, *et al.* Ty elements transpose through an RNA intermediate. *Cell* 1985;**40**(3):491–500.
12. Bourque G, Burns KH, Gehring M, *et al.* Ten things you should know about transposable elements. *Genome Biol*, **19**(1):199, Dec 2018.
13. Orozco-Arias S, Isaza G, Guyot R. Retrotransposons in plant genomes: structure, identification, and classification through bioinformatics and machine learning. *Int J Mol Sci* 2019;**20**(15):3837.
14. Neumann P, Novák P, Hoštáková N, *et al.* Systematic survey of plant ltr-retrotransposons elucidates phylogenetic relationships of their polyprotein domains and provides a reference for element classification. *Mobile DNA* 2019;**10**(1):1–17.
15. Larrañaga P, Calvo B, Santana R, *et al.* Machine learning in bioinformatics. *Brief Bioinform* 2006;**7**(1):86–112.
16. Mjolsness E, DeCoste D. Machine learning for science: state of the art and future prospects. *Science* 2001;**293**(5537):2051–5.
17. Janiesch C, Zschech P, Heinrich K. Machine learning and deep learning. *Electron Markets* 2021;**31**(3):685–95.
18. Zou J, Huss M, Abid A, *et al.* A primer on deep learning in genomics. *Nat Genet* 2019;**51**(1):12–8.
19. Greener JG, Kandathil SM, Moffat L, *et al.* A guide to machine learning for biologists. *Nat Rev Mol Cell Biol* 2022;**23**(1):40–55.
20. Loureiro T, Camacho R, Vieira J, *et al.* Improving the performance of transposable elements detection tools. *J Integr Bioinform* 2013;**10**(3):231.
21. Nakano FK, Mastelini SM, Barbon S, *et al.* Improving hierarchical classification of transposable elements using deep neural networks. In: *Proceedings of the International Joint Conference on Neural Networks*, Vol. **8–13 July**. Rio de Janeiro: IEEE, 2018.
22. Panta M, Avdesh Mishra M, Hoque T, *et al.* Classifyte: a stacking-based prediction of hierarchical classification of transposable elements. *Bioinformatics* 2021;**37**:2529–36.
23. Orozco-arias S, Piña JS, Tabares-soto R, *et al.* Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements. *Processes* 2020;**8**(638):1–20.
24. Orozco-Arias S, Candamil-cortés MS, Jaimes PA, *et al.* K -mer-based machine learning method to classify LTR-retrotransposons in plant genomes. *Peer J* 2021;**9**:e11456.
25. Girgis HZ. Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on the genomic scale. *BMC Bioinformatics* 2015;**16**(1):1–19.
26. Hoede C, Arnoux S, Moisset M, *et al.* PASTEC: an automatic transposable element classification tool. *PLoS ONE* 2014;**9**(5):1–6.
27. Abrusán G, Grundmann N, Demester L, *et al.* TEclass - a tool for automated classification of unknown eukaryotic transposable elements. *Bioinformatics* 2009;**25**(10):1329–30.
28. Riehl K, Riccio C, Miska EA, *et al.* Transposonultimate: software for transposon classification, annotation and detection. *Nucleic Acids Res* 2022;**50**:gkac136.
29. Rawal K, Ramaswamy R. Genome-wide analysis of mobile genetic element insertion sites. *Nucleic Acids Res* 2011;**39**(16): 6864–78.
30. Jiang SY, Ramachandran S. Genome-wide survey and comparative analysis of LTR retrotransposons and their captured genes in rice and sorghum. *PLoS ONE* 2013;**8**(7):e71118.
31. Schietgat L, Vens C, Cerri R, *et al.* A machine learning based framework to identify and classify long terminal repeat retrotransposons. *PLoS Comput Biol*, **14**(4):e1006097, Apr 2018.
32. Zhao X, Wang H. LTR-FINDER: an efficient tool for the prediction of full-length LTR retrotransposons. *Nucleic Acids Res* 2007;**35**(SUPPL.2):265–8.
33. Shujun O, Weija S, Liao Y, *et al.* Benchmarking transposable element annotation methods for creation of a streamlined, comprehensive pipeline. *Genome Biol* 2019;**20**(1):1–18.
34. Orozco-Arias S, Liu J, Tabares-Soto R, *et al.* Inpactor, integrated and parallel analyzer and classifier of LTR retrotransposons and ITS application for pineapple LTR retrotransposons diversity and dynamics. *Biology* 2018;**7**(2):32–48.
35. Mhiri C, Borges F, Grandbastien M-A. Specificities and dynamics of transposable elements in land plants. *Biology* 2022;**11**(4):488.
36. Arkhipova IR. Using bioinformatic and phylogenetic approaches to classify transposable elements and understand their complex evolutionary histories. *Mobile DNA* 2017;**8**(1):1–14.
37. Bao W, Kojima KK, Kohany O. Repbase update, a database of repetitive elements in eukaryotic genomes. *Mobile DNA* 2015;**6**(1):4–9.

38. Amselem J, Cornut G, Choisne N, *et al.* RepetDB: a unified resource for transposable element references. *Mobile DNA* 2019;**10**(1):1–8.
39. Spannagl M, Nussbaumer T, Bader KC, *et al.* PGSB plantsDB: updates to the database framework for comparative plant genome research. *Nucleic Acids Res* 2016;**44**(D1):D1141–7.
40. Orozco-Arias S, Jaimes PA, Candamil MS, *et al.* InpactorDB: a classified lineage-level plant LTR retrotransposon reference library for free-alignment methods based on machine learning. *Genes* 2021;**12**(2):1–17.
41. Montesinos-López OA, Montesinos-López A, Pérez-Rodríguez P, *et al.* A review of deep learning applications for genomic selection. *BMC Genomics* 2021;**22**(1):1–23.
42. da Cruz MHP, Saito PTM, Paschoal AR, *et al.* Classification of transposable elements by convolutional neural networks. In: *Lecture Notes in Computer Science*, Vol. **11509**. Zakopane, Poland: Springer International Publishing, 2019, 157–68.
43. Pereira da Cruz MH, Domingues DS, Saito PTM, *et al.* TERL: classification of transposable elements by convolutional neural networks. *Brief Bioinform* 2021;**22**(3):bbaa185.
44. Yan H, Bombarely A, Li S. DeepTE: a computational method for de novo classification of transposons with convolutional neural network. *Bioinformatics (Oxford, England)* 2020;**36**:4269–75.
45. Zhang R-G, Wang Z-X, Shujun O, *et al.* Tesorter: lineage-level classification of transposable elements using conserved protein domains. bioRxiv. 2019.
46. Llorens C, Futami R, Covelli L, *et al.* The gypsy database (gydb) of mobile genetic elements: release 2.0. *Nucleic Acids Res* 2010;**39**(suppl_1):D70–4.
47. Jiang N, Bao Z, Zhang X, *et al.* An active DNA transposon family in rice. *Nature* 2003;**421**(6919):163–7.
48. Jiang N, Bao Z, Zhang X, *et al.* Pack-MULE transposable elements mediate gene evolution in plants. *Nature* 2004;**431**(7008):569–73.
49. Feschotte C, Swamy L, Wessler SR. Genome-wide analysis of mariner-like transposable elements in rice reveals complex relationships with Stowaway miniature inverted repeat transposable elements (MITEs). *Genetics* 2003;**163**(2):747–58.
50. Xie Y, Wang Y, Ray W. A rice DNA sequence that resembles the maize Mu 1 transposable element. *Rice Genetics Collect* 2008;**2**:377–87.
51. Barret P, Brinkman M, Beckert M. A sequence related to rice Pong transposable element displays transcriptional activation by in vitro culture and reveals somaclonal variations in maize. *Genome* 2006;**49**(11):1399–407.
52. Smit AFA, Hubley R, Green P. Repeatmasker open-4.0. 2015; 2013–5.
53. Shujun O, Jiang N. LTR_FINDER_parallel: parallelization of LTR_FINDER enabling rapid identification of long terminal repeat retrotransposons. *Mobile DNA* 2019;**10**(1):1–3.
54. Raharimalala N, Rombauts S, McCarthy A, *et al.* The absence of the caffeine synthase gene is involved in the naturally decaffeinated status of *Coffea humblotiana*, a wild species from Comoro archipelago. *Sci Rep* 2021;**11**(1):1–14.
55. Zhichao X, Xiangdong P, Gao R, *et al.* Tandem gene duplications drive divergent evolution of caffeine and crocin biosynthetic pathways in plants. *BMC Biol* 2020;**18**(1):1–14.
56. Shujun O, Jiang N. LTR_retriever: a highly accurate and sensitive program for identification of long terminal repeat retrotransposons. *Plant Physiol* 2018;**176**(2):1410–22.
57. Ellinghaus D, Kurtz S, Willhoeft U. LTRharvest, an efficient and flexible software for de novo detection of LTR retrotransposons. *BMC Bioinformatics* 2008;**9**:1–14.
58. Redmon J, Divvala S, Girshick R, *et al.* You only look once: unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, USA: IEEE, 2016, 779–88.
59. Rodriguez M, Makałowski W. Software evaluation for de novo detection of transposons. *Mobile DNA* 2022;**13**(1):1–14.
60. This P, Lacombe T, Cadle-Davidson M, *et al.* Wine grape (*Vitis vinifera* L.) color associates with allelic variation in the domestication gene vvmyba1. *Theor Appl Genet* 2007;**114**(4):723–30.
61. Xiao H, Jiang N, Schaffner E, *et al.* A retrotransposon-mediated gene duplication underlies morphological variation of tomato fruit. *Science* 2008;**319**(5869):1527–30.
62. Momose M, Abe Y, Ozeki Y. Miniature inverted-repeat transposable elements of stowaway are active in potato. *Genetics* 2010;**186**(1):59–66.
63. Butelli E, Licciardello C, Zhang Y, *et al.* Retrotransposons control fruit-specific, cold-dependent accumulation of anthocyanins in blood oranges. *Plant Cell* 2012;**24**(3):1242–55.
64. Wei L, Cao X. The effect of transposable elements on phenotypic variation: insights from plants to humans. *Sci China Life Sci* 2016;**59**(1):24–37.
65. Bonchev G, Parisod C. Transposable elements and microevolutionary changes in natural populations. *Mol Ecol Resour* 2013;**13**(5):765–75.
66. Li S-F, Ting S, Cheng G-Q, *et al.* Chromosome evolution in connection with repetitive sequences and epigenetics in plants. *Genes* 2017;**8**(10):290.
67. Shujun O, Chen J, Jiang N. Assessing genome assembly quality using the LTR assembly index (lai). *Nucleic Acids Res* 2018;**46**(21):e126–6.
68. Casacuberta E, González J. The impact of transposable elements in environmental adaptation. *Mol Ecol* 2013;**22**(6):1503–17.
69. Loureiro T, Camacho R, Vieira J, *et al.* Boosting the detection of transposable elements using machine learning. In: *7th International Conference on Practical Applications of Computational Biology & Bioinformatics*. Salamanca, Spain: Springer, 2013, 85–91.
70. Santos BZ, Pereira GT, Nakano FK, *et al.* Strategies for selection of positive and negative instances in the hierarchical classification of transposable elements. In: *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*. São Paulo, Brazil: IEEE, 2018, 420–5.
71. Pandey P, Bender MA, Johnson R, *et al.* Squeakr: an exact and approximate k-mer counting system. *Bioinformatics* 2018;**34**(4):568–75.
72. Flutre T, Permal E, Quesneville H. Transposable element annotation in completely sequenced eukaryote genomes. In: *Plant Transposable Elements*. Springer, 2012, 17–39.
73. Zhou S-S, Yan X-M, Zhang K-F, *et al.* A comprehensive annotation dataset of intact ltr retrotransposons of 300 plant genomes. *Sci Data* 2021;**8**(1):1–9.