Bioinformatics, 36(15), 2020, 4269–4275
doi: 10.1093/bioinformatics/btaa519
Advance Access Publication Date: 16 May 2020
Original Paper

OXFORD

Sequence analysis

DeepTE: a computational method for *de novo* classification of transposons with convolutional neural network

Haidong Yan[1], Aureliano Bombarely[1,2] and Song Li [1,3,*]

[1]School of Plant and Environmental Sciences (SPES), Virginia Tech, Blacksburg, VA 24061, USA, [2]Department of Life Sciences, University of Milan, Milan 20122, Italy and [3]Graduate Program in Genetics, Bioinformatics and Computational Biology (GBCB), Virginia Tech, Blacksburg, VA 24061, USA

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on January 9, 2020; revised on April 12, 2020; editorial decision on May 9, 2020; accepted on May 12, 2020

Abstract

Motivation: Transposable elements (TEs) classification is an essential step to decode their roles in genome evolution. With a large number of genomes from non-model species becoming available, accurate and efficient TE classification has emerged as a new challenge in genomic sequence analysis.

Results: We developed a novel tool, DeepTE, which classifies unknown TEs using convolutional neural networks (CNNs). DeepTE transferred sequences into input vectors based on *k*-mer counts. A tree structured classification process was used where eight models were trained to classify TEs into super families and orders. DeepTE also detected domains inside TEs to correct false classification. An additional model was trained to distinguish between non-TEs and TEs in plants. Given unclassified TEs of different species, DeepTE can classify TEs into seven orders, which include 15, 24 and 16 super families in plants, metazoans and fungi, respectively. In several benchmarking tests, DeepTE outperformed other existing tools for TE classification. In conclusion, DeepTE successfully leverages CNN for TE classification, and can be used to precisely classify TEs in newly sequenced eukaryotic genomes.

Availability and implementation: DeepTE is accessible at https://github.com/LiLabAtVT/DeepTE.

Contact: songli@vt.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Transposable elements (Transposons; TEs) constitute a large portion of many known eukaryotic genomes (Makałowski, 2001; SanMiguel *et al.*, 1996), and play significant roles in many biological processes (Bourque *et al.*, 2018). Accurate identification and annotation of TEs are essential to the understanding of their roles in genome evolution, genome stability and regulation of gene expression (Goerner-Potvin and Bourque, 2018; Platt *et al.*, 2016; Wicker *et al.*, 2007). With the reduced sequencing cost and novel long-read sequencing technologies, a large number of eukaryotic genomes have been sequenced in recent years. Given the diversity and abundance of TEs, annotation and classification of TEs have reemerged as major challenges in genome annotation (Platt *et al.*, 2016).

TEs fall into two general categories (Wicker *et al.*, 2007). Class I TEs are retrotransposons, which can transpose from one to another position via a 'copy-and-paste' mechanism. This group contains long terminal repeat (LTR) retroelements, such as Gypsy and Copia, and non-LTR retroelements, such as dictyostelium intermediate repeat sequence (DIRS), penelope-like elements (PLE), long interspersed nuclear element (LINE) and short interspersed nuclear element (SINE). Class II TEs are DNA transposons following a 'cut-and-paste' mechanism, characterized by terminal inverted repeats (TIR). In Subclass 1 of Class II (Class II_sub1) TEs, nine known super families can be distinguished by target site duplication (TSD) and TIR sequences. These TEs can be further classified into autonomous and non-autonomous elements based on their abilities to move by themselves. Miniature inverted repeat transposable element (MITE) is a special type of non-autonomous Class II_sub1 TEs with higher copy numbers and special structural feature and do not encode any transposase (TR). Subclass 2 of Class II (Class II_sub2) TEs are DNA TEs that undergo a transposition process without double-stranded cleavage. Helitron and Maverick are two major orders in this group (Wicker *et al.*, 2007).

In the past, several computational tools have been developed to identify TEs without classifying TEs into super families. For

© The Author(s) 2020. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022

example, LTR TEs (LTRs) can be identified by LTR_STRUC (McCarthy and McDonald, 2003), MGEScan (Rho *et al.*, 2007), LTR_FINDER (Xu and Wang, 2007) and LTRharvest (Ellinghaus *et al.*, 2008). But these four tools cannot classify LTRs into super families, such as Copia or Gypsy. MITE TEs (MITEs) can be identified by MITE-Hunter (Han and Wessler, 2010), detectMITE (Ye *et al.*, 2016), MITEFinderII (Hu *et al.*, 2018) and MITE Tracker (Crescente *et al.*, 2018). However, MITEs identified from these four tools are not classified into super families under TIR order (Wicker *et al.*, 2007). RepeatModeler is able to build and classify consensus models of putative interspersed repeats, based on sequence similarity to known repeats (Smit and Hubley, 2008). When we tested this method using in maize 'B73' genome (Schnable *et al.*, 2009), ~14% TEs are unclassified.

Several software packages have been developed for TE classification including TECLASS (Abrusán *et al.*, 2009), REPCLASS (Feschotte *et al.* 2009) and PASTEC (Hoede *et al.*, 2014). TECLASS leverages Support Vector Machine to classify TEs into Class I, Class II, LINE and SINE, but only these four groups are classified. REPCLASS utilizes structural and homology characterization modules to classify TEs into more groups than TECLASS, such as Class I and II, super families of LTRs, DNA TEs, LINE, SINE and Helitron. REPCLASS depends on WU-BLAST, which is an alignment software that is no longer maintained. An additional shortcoming is that both TECLASS and REPCLASS cannot distinguish between TEs and other non-TE sequences (non-TEs). REPET is a bioinformatics pipeline for identifying and annotating TEs in genome sequences (Flutre *et al.*, 2011; Quesneville *et al.*, 2005). REPET can use PASTEC in the classification step of the pipeline. PASTEC uses Hidden Markov Model profiles to classify TEs based on conserved functional domains of the proteins, and it identifies more super families than REPCLASS. Performance of PASTEC was shown to be better than TECLASS and REPCLASS, and it could distinguish potential host genes, rDNA and SRR sequences from real TEs. However, the sensitivity (SE) in detecting certain groups of TEs, such as TIR (64.1%), LTR (39.1%) and Class I (53.0%) groups are low, and it cannot distinguish super families under TIR, LINE and SINE orders. Additional methods, such as LTRdigest (Steinbiss *et al.*, 2009) and LTR_retriever (Ou and Jiang, 2018) can classify LTR elements, and TIR-learner can classify TIR elements (Su *et al.*, 2019). But these three tools cannot simultaneously classify unknown TEs from either Class I or Class II.

In this article, we have developed DeepTE, a deep-learning method to classify TEs from non-TEs and to classify TEs into multiple orders and classes. Deep learning has been widely applied and has become one effective strategy in identifying complex patterns derived from feature-rich datasets. One particular deep-learning model—CNN—has achieved outstanding performance in image classification, speech recognition and natural language processing (Krizhevsky *et al.*, 2012; Schmidhuber, 2015). CNN model has also been successfully applied in prediction of unknown sequence profiles or motifs and functional activity discovery, without pre-defining sequence features. For example, CNN has been used in prediction of sequence specificities of DNA- and RNA-binding proteins (Alipanahi *et al.*, 2015), effects of non-coding variants (Zhou and Troyanskaya, 2015) and classification of alignments of non-coding RNA sequences (Alipanahi *et al.*, 2015; Aoki and Sakakibara, 2018; Schmidhuber, 2015; Zeng *et al.*, 2016; Zhou and Troyanskaya, 2015).

Traditionally, transposons can be classified into different super families underlying distinct sequence patterns (Wicker *et al.*, 2007). In view of this, we have designed DeepTE to classify TEs into seven orders which include 15, 24 and 16 super families in plants, metazoans and fungi, respectively. DeepTE also combines TE domain detection, which improves its performance, and outperforms PASTEC in most TE categories.

2 Materials and methods

2.1 Transposon dataset

DeepTE was built using RepBase (Bao *et al.*, 2015) and Plant Genome and Systems Biology (PGSB) repeat database (Spannagl *et al.*, 2016). Repbase contains prototypic sequences representing repetitive DNA from 134 species. PGSB database provides access to the repeat element database from 44 species and covers 20 different genera. We combined these two datasets since TEs in these two databases are largely not overlapping. We found 74 383 TEs when combining sequences from both databases. After removing identical TEs using BLAST (Madden, 2013), there were 71 049 unique TEs in the dataset that was used for model training and testing. Species-specific (plants, metazoans and fungi) training data were selected based on the presence of TE families in different species. For example, 63 416 TEs (out of 71 049 TEs) were used for plants (see Supplementary Table S1 for a detailed breakdown of numbers of TEs in each family in plants). Because both databases only contain Helitron, the plant Class II_sub2 classifier only represents Helitron superfamily. To classify Class II_sub1 transposons into MITE and nMITE (not MITE) TEs, we manually created MITE and nMITE datasets. Class II_sub1 TEs annotated as MITE in RepBase and PGSB were labeled as 'MITE'. The rest of Class II_sub1 TEs were annotated for PfamA domains with HMMER3 (Eddy, 2010). The TEs were defined as 'nMITE' where TR domains were identified with length ≥800 bp (Wicker *et al.*, 2007) (Supplementary Table S2). A total of 752 TEs were defined as 'MITE', and 2696 TEs were defined as 'nMITE'.

2.2 Transposon sequence representation

We used *k*-mer occurrence to construct feature matrices from the TE sequences. This system has been successfully applied to bacteria taxonomic classification of metagenomic data (Fiannaca *et al.*, 2018). The number of features was set to $L=4^k$, where k is the length of the *k*-mers (Fig. 1A). A sliding window over the input vector was used to generate the convolution operation of the first stage of the CNN network. In our analysis, we used $k=3$ to 7 to test the performance of different *k*-mer sizes.

2.3 CNN classifier

We used the python package Keras (version 2.2.4) (Chollet, 2015) to implement CNN in our model. Our neural network was consisted of three hidden layers with kernel sizes of three and three max-pooling layers with pool size of two (Fig. 1B). The rationale of using CNN and the reasons for the choices of hyper-parameters were included in Section 4. Hidden layers resided in-between input and output layers. For each input sequence, *k*-mer frequency was

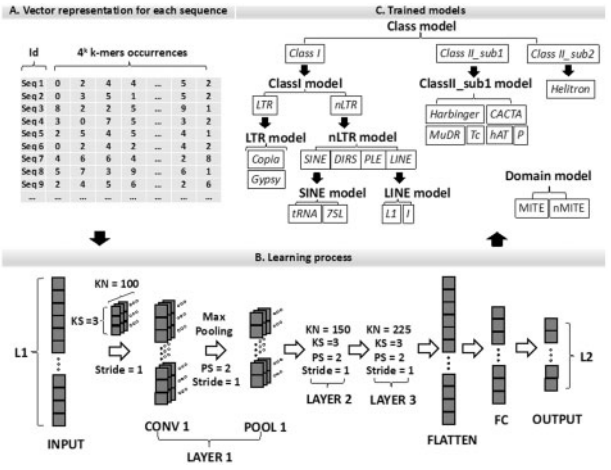

Fig. 1. Training process of neural network. (**A**) The training sequences were converted to a matrix of *k*-mers occurrences. (**B**) The architecture of the CNN. *L*1 and *L*2 indicate the dimension of the input and the output vectors. KS: kernel size. KN: number of kernels. PS: max-pooling window size. FC: fully connected layer with 128 units. LAYERs 2 and 3 had similar architectures as LAYER 1 except for number of kernels. (**C**) Eight models (bold fonts) were trained in our classification pipeline. This schematic diagram shows which model can classify which types of TEs. For example, **Class model** classifies TEs into Class I, Class II_sub1 and Class II_sub2; and **ClassI model** classifies TEs into LTR and nLTR

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022

calculated for different sizes of k (k=3 to 7). In the input layer, k-mers were ordered alphabetically and the orders were identical for all input sequences. Kernel is a small vector of size KS (KS=3) where KS adjacent k-mer-frequencies are used as input for a neuron. We set the stride size as one, which means the kernel window is moving along the input vector with a step size of one. Each convolution layer consisted of KN (KN=100, 150 and 225) number of kernels. Max pooling is used to transform data by taking the maximum value from the values in the window and max-pooling size represents the length of the pooling window. Max pooling was used to summarize the output of each two adjacent kernels with a stride of one. A dropout value of 0.5 was set after the last convolutional layer. The dropout is the probability of training a given node in a layer (0 means no output from this layer, and 1.0 means no dropout). The output of max-pooling layer was flattened as input for the next convolution layer. After the max-pooling layer of the third convolution layer, a fully connected layer was set to densely connect the flattened max-pooling layer to 128 units. A dropout value of 0.5 was set after this layer. A softmax output layer was set to compute the probabilities for the classes of input sequences (Fig. 1B). ReLU function was used in all three hidden layers and the fully connected layer. ReLU is an activation function that introduces non-linear properties to the network, and converts an input signal of a node to an output signal. ReLU gives an output x if x is positive and 0 otherwise (Agarap, 2018). We used the categorical_crossentropy loss function, and ADAM optimizer with a learning rate of 0.001 (Kingma and Ba, 2014).

2.4 Improving classification via detecting transposon conserved domains

The eight models were organized according to TE classification system (Wicker *et al.*, 2007) (Supplementary Fig. S1). To improve the performance of classification, conserved domains from TE sequences were identified using PfamA domains with HMMER3 (Eddy, 2010) (Supplementary Table S2). Classification results from **Class** model and **ClassI** model were corrected by the following criteria: (i) if TEs were classified into Class I group with domain 'TR', the predicted 'Class I' was changed to 'Class II_sub1' in the **Class** model; (ii) if TEs were classified into Class II_sub1 group with domain Reverse Transcriptase, this prediction was changed to 'Class I' in the **Class** model; (iii) if TEs were classified into LTR but with Endonuclease ('EN') domain, this predicted group was corrected to 'nLTR', since 'EN' is exclusive in the nLTR TEs (Wicker *et al.*, 2007) (Supplementary Fig. S1).

2.5 Model training and evaluation

The dataset was divided into training (95%) and testing (5%) sets for each TE family. We performed 10-fold cross validation in the training dataset, and separated the training set to sub-training (90%) and validation (10%) sets. The sub-training and validation sets were used to compare performance of different k-mer sizes using precision (PR), SE and $F1$-score ($F1$) for eight models. These eight models were then used to classify TEs from the testing set (Supplementary Fig. S1). To further evaluate the performance of DeepTE, we compared DeepTE with the latest TE classification tool called PASTEC (Hoede *et al.*, 2014). True positive (TP), true negative (TN), false positive (FP) and false negative (FN) TE number were calculated and SE, specificity (SP), accuracy (AC), PR, geometric mean (GM) of SE and SP and Matthews Correlation Coefficient (MCC) were evaluated. These scores were calculated for the test TEs using PASTEC software with default settings and compared with DeepTE. SE, SP, AC, PR, GM, MCC and $F1$ were defined as follows:

$$\begin{gathered}
SE = TP/(TP+FN);\ SP = TN/(FP+TN);\ PR = TP/(TP+FP);\\
GM = \sqrt{SE \times SP};\ AC = (TP+TN)/(TP+TN+FP+FN);\\
MCC = (TP \times TN - FP \times FN)/\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}\\
F1 = 2 \times (PR \times SE/(PR+SE)).
\end{gathered}$$

Tukey's honestly significant difference test is used as a conservative statistical test to find significant differences in all pairwise comparisons and to control for family-wise error rate (Abdi and Williams, 2010).

2.6 Classifying non-transposon and transposon sequences

To differentiate non-TEs from TE sequences, we collected non-transposon sequences including coding sequences (CDS) and intergenic sequences (INS). The CDS from 21 plant species were collected from phytozome (https://phytozome.jgi.doe.gov/index.html) and Sol Genomics Network (https://solgenomics.net/) (Supplementary Table S3). Sequences of CDS from each species were combined and CDS annotated with transposons were removed. From these CDS sequences, we randomly selected ~800 000 sequences as our CDS dataset. These sequences were then randomly divided into 10 datasets of each contained ~80 000 sequences. For INS, 16 plant species with known repeat annotations in the phytozome were analyzed. The INS of each species without overlapping with transposons were collected, and sequences with length between 50 and 10 000 bp were retained. ~800 000 sequences from the filtered sequences were selected, and randomly divided into 10 replicated datasets with ~80 000 for each. Each set from CDS and INS was separated into training (90%) and validation (10%) sets. TE dataset was constructed using all TE sequences (~80 000). These TE sequences were randomly partitioned into 10 equal sized subsamples, and a single subsample was retained as the validation set and the remaining 9 subsamples were used as training set. This process was repeated 10 times to obtain 10 replicated TE datasets with each containing training and validation sets. For replicates in CDS, INS and TE datasets, we selected one replicate from each of these datasets, and combined them to one final replicate. This process was repeated 10 times and each replicate was used exactly once. A total of 10 final replicates were generated to train model. Three scores (PR; SE; $F1$) generated from keras packages were used to evaluate performance of this model.

3 Results

3.1 Performance comparison for different k-mer sizes

To identify suitable k-mer size for representing the input sequences, performances of different k-mer sizes were compared. The PR changed as the k-mer size changed in all eight trained models (Fig. 2). The performance improved significantly (P-value <0.05) with increased k-mer sizes in **Class**, **ClassI**, **ClassII_sub1**, **LTR** and **nLTR** models. No significant change was found in **Domain**, **LINE** and **SINE** models. Using k=6 had similar performance with using k=7 for most models except for **ClassII_sub1** model. In model **ClassI** and **nLTR**, using k=5, 6 and 7 had the same performance. Using k=3 had the lowest PR across all tested k-mer sizes. Similar results were obtained using SE and $F1$ (Supplementary Fig. S2). Taken together, using k=7 had the best performance in all models, and it was used for further analysis. Using larger k-mer sizes requires substantial more computing time and was not tested in this analysis.

3.2 Performance of trained models

To further evaluate performance from these eight models, PR, SE and $F1$s, were calculated (Table 1). In **Class** model, these three scores are higher than 0.90 for Class I and Class II_sub1 TEs. For Class II_sub2 (Helitron) TEs, although PR (0.88) is close to 0.90, its SE (0.44) and $F1$ (0.59) are the lowest among these three classifiers. In **ClassI** model, LTR and nLTR both have high performance, and the performance scores are all higher than 0.97 for LTRs. **ClassII_sub1** model can classify six DNA/TE families, with variable performances. In P super family, PR (0.50), SE (0.10) and $F1$ (0.18) are lower than other families. Mutator and Harbinger have higher PR than SE and $F1$. For TcMar and hAT, their SE (>0.80) is higher than PR (<0.80) and $F1$ (<0.80). CACTA has relative consistent scores with average of 0.70. For **LTR**, **nLTR** and **LINE** models, all super families have similar performance scores, and most scores are above 0.90. Particularly, **Domain** model displays the best

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022

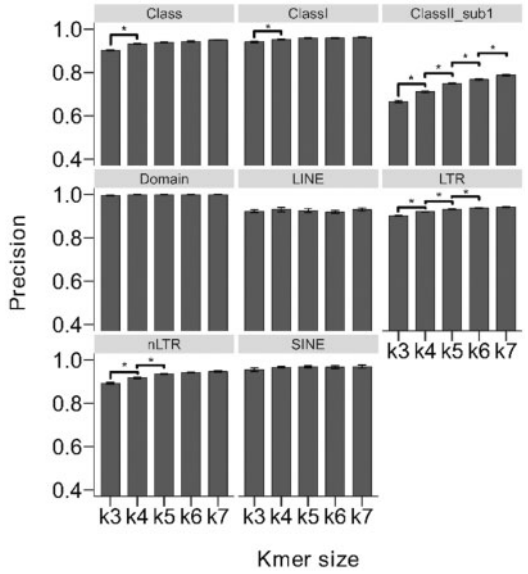

Fig. 2. PR validation of CNN classifier based on *k*-mer size. *k*3, *k*4, *k*5, *k*6 and *k*7 represent *k*-mer size from 3 to 7. Asterisks between two nearby variables indicate significant difference with *P*-value <0.05

Table 1. Performance of eight models in plants

Model	Group	PR[a]	SE[b]	*F*1[c]
Class (moderate)	Class I	0.92	0.93	0.93
	Class II_sub1	0.91	0.94	0.93
	Class II_sub2	0.88	0.44	0.59
ClassI (high)	LTR	0.98	0.97	0.97
	nLTR	0.91	0.94	0.93
ClassII_sub1 (poor)	TcMar	0.66	0.84	0.74
	hAT	0.76	0.82	0.79
	Mutator	0.85	0.60	0.71
	P	0.50	0.11	0.18
	Harbinger	0.83	0.60	0.70
	CACTA	0.73	0.71	0.72
LTR (good)	Copia	0.95	0.89	0.92
	Gypsy	0.93	0.97	0.95
nLTR (good)	LINE	0.98	0.94	0.96
	SINE	0.97	0.97	0.97
	DIRS	0.83	0.95	0.88
	PLE	0.88	0.92	0.90
LINE (high)	L1	0.96	1.00	0.98
	I	1.00	0.90	0.95
SINE (high)	tRNA	1.00	1.00	1.00
	7SL	1.00	1.00	1.00
Domain (high)	MITE	0.99	1.00	0.99
	nMITE	1.00	1.00	1.00

[a]Precision.
[b]Sensitivity.
[c]*F*1-score.

performance with MITEs and nMITEs achieved to maximal value for these three scores (Table 1).

In summary, we can roughly divide the models into four categories: high, good, moderate and poor performance (Table 1). High performance models have all three scores higher than or equal to 0.9. Good models have at least one score higher than or equal to 0.9 and all scores are higher than 0.7. Moderate models have no more than two scores that are lower than 0.7. Poor models have all three scores below 0.7. Among all TE classification results, we have 50% high-performance models (4/8), 25% good models (2/8), one moderate model and one poor model. We compared the *F*1 of the training and testing datasets and we found that all models showed <5% divergence in the training and testing *F*1 except for ClassII_sub1 (Supplementary Fig. S3). This result suggests that overfitting is limited in our models.

3.3 Improving classification by discovering conserved domains of transposon sequences

These eight models were wrapped together to classify unknown TEs and the classification performance were evaluated with four scores including AC, PR, SE and SP (Supplementary Figs S1 and S4). DeepTE showed high AC over 0.90 for all TE groups, and more than half of them (61%; 14/23) had almost perfect AC over 0.98 (Supplementary Fig. S4A). In Class I, all nLTR groups showed higher AC than all LTR groups. Among all TE groups, 70% (16/23) had SE higher than 0.80, and most Class I groups (92%; 12/13) had SE over 0.80. ClassII_sub1_DNA_P showed the least SE (0), probably due to low number of TEs in the dataset (Supplementary Table S1). ClassI_nLTR_SINE_7SL, MITEs and nMITEs had the highest SE (Supplementary Fig. S4B). For PR, over half of TE groups (57%; 13/23) were over 0.70 (Supplementary Fig. S4C), while for SP, nearly all groups were over 0.90 (Supplementary Fig. S4D). Taken together, our method showed higher SE for Class I groups than Class II_sub1 and Class II_sub2 groups. For AC, PR andSP, our method showed similar performance in all TE groups. Remarkably, for Class II_sub1 MITE and nMITE groups, our method generated SE, AC, PR and SP close to 100%. Our method showed lower SE for Class II_sub2 than the other TE groups, but it performed better in AC, PR and SP for Class II_sub2.

We can also roughly divide the 23 TE groups into four categories as defined in the performance of trained models: high, good, moderate and poor. In total, we have 4 high-performance models (4/23), 8 good models (8/23) and 11 moderate models (11/23).

TE families were defined with high DNA sequence similarity for their typical internal domain or coding region (Wicker *et al.*, 2007). To improve performance of TE classification in DeepTE, conserved domains of TEs were detected to correct false classification (Supplementary Fig. S1 and Table S2). Several groups showed slightly improved AC, SE, PR and SP; however, these improvements are not statistically significant (Supplementary Fig. S5 and Table S4).

In general, these eight models achieved good performance, particular for AC and SP, and it had relative better SE in Class I, than in Class II_sub1 and II_sub2 families. The classification performance was slightly increased after correction of false prediction *via* detecting domains within TE sequences.

3.4 Performance comparison between DeepTE and PASTEC

DeepTE was compared with a published TE classifier called PASTEC. There are 11 TE groups that can be classified by both DeepTE and PASTEC, and multiple scores for model performances were used in our comparison. GM (Ding *et al.*, 2010) and MCC (Silva *et al.*, 2019) have been suggested as ways to evaluate model performance and are less sensitive to imbalanced dataset than other commonly used metrices. In DeepTE, most groups had GM score over 0.90 (Fig. 3A) and MCC score over 0.80 (Supplementary Fig. S6), except for Class II_sub2 group which displayed lower GM (0.64) and MCC (0.60). In PASTEC, all groups showed significantly (*P*-value < 0.05) lower MCC and GM than in DeepTE, and no TE was detected in DIRS, PLE and Class II_sub2 (Fig. 3A and Supplementary Fig. S6A).

PASTEC showed lower SE in all TE groups (Fig. 3B), of which the highest SE was smaller than 0.65. In DeepTE, all groups had SE higher than 0.85, except for Class II_sub2 (SE=0.42) (Fig. 3B). In comparison of AC, PASTEC showed lower performance as compared to DeepTE for most groups, except for Class II_sub1, Class II_sub2 and PLE where PASTEC showed no significant difference than DeepTE. DeepTE had more than 2-folds AC than PASTEC in Class I, and more than 1.5-folds AC in LTR (Supplementary Fig. S6B). As compared to DeepTE, PASTEC had higher PR in four TE

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022

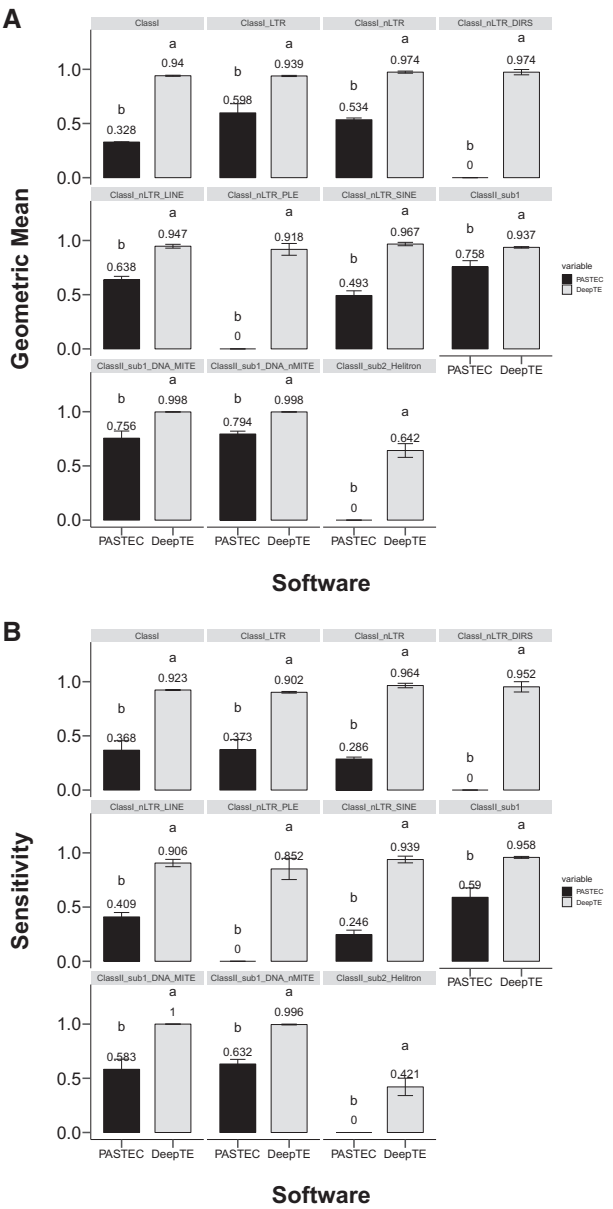

Fig. 3. Performance comparison between DeepTE and PASTEC. All pairwise comparisons are statistically significant as indicated by different letters (a, b)

groups, lower PR in four TE groups and no significant difference in three TE groups. The PR in DIRS, PLE and Class II_sub2 was 0 in PASTEC (Supplementary Fig. S6C). For SP, seven groups in PASTEC displayed better performance than DeepTE, except for Class I in DeepTE with more than 3-folds SP than PASTEC. No significant difference was found in MITE, nMITE and Class II_sub2 (Supplementary Fig. S6D). In general, PASTEC detected low number of FPs, resulting in better PR than DeepTE in LTR, nLTR, LINE, SINE and Class II_sub1 groups. But it failed to detect many of TP TEs, and it did not find any TPs in DIRS, PLE and Class II_sub2. The improved performance of DeepTE over PASTEC is also verified using area under the receiver operating characteristic curve (Supplementary Fig. S6) and area under the precision-recall curve (Supplementary Fig. S7) (Kamath *et al.*, 2014). Taken together, we conclude that DeepTE outperforms PASTEC.

We also compared the speed for DeepTE and PASTEC. The computing time for DeepTE and PASTEC were tested on a Linux workstation with Inter(R) Xeon(R) Gold 5115 CUP (2.40 GHz), eight cores and 40 GB memory. There is one graphic card installed in this Linux workstation: NVIDIA Corporation GP102GL [Tesla P40] (1.30 GHz base and 1.53 GHz as boosters) with 24 GB memory. For DeepTE, the time used to classify 1084 TEs was 71±3 s, and for PASTEC, the time used to classify these TEs was 1305±7 s. These results suggest that DeepTE is 18.3 times faster than PASTEC when classifying TEs.

3.5 Combined training data from two databases have better performance

Data from RepBase and PGSB database were combined to train the DeepTE models. We further compared of models trained on combined RepBase and PGSB data (RB_PG models) and models trained by data only from RepBase (RB models). The testing dataset was divided into RepBase and PGSB datasets (see Supplementary Table S5 for number of TEs used in testing datasets). In both RepBase (Supplementary Fig. S9) and PGSB (Supplementary Fig. S10) testing datasets, the RB_PG models achieved better performance than the RB models in both LTR families and five Class II_sub1 families. These results suggest classification performance would improve by using training datasets from both RepBase and PGSB database.

3.6 Classification of non-TE and TE sequences in plants

As an additional function for DeepTE, we have trained a model to classify CDS, INS and TE datasets in plants to differentiate non-TE and TE sequences. Three scores (PR, SE and *F*1) were used to evaluate model performance. The *F*1 in CDS was significantly (*P*-value <0.05) higher than TE and INS groups, and no difference was found between TE and INS groups. For PR, score in INS was lower than scores in TE and CDS groups. CDS showed the highest score in SE, while the score in TE was the lowest (Fig. 4). In total, CDS group achieved the best performance comparing with TE and INS groups.

3.7 TE classification in metazoans and fungi

Metazoans and fungi have TE super families that are not detected in plants, we developed another two training datasets containing 71 049 and 63 449 TEs in metazoans (Supplementary Table S7) and fungi (Supplementary Table S8), respectively. A total of 7 models were trained to classify TEs into 24 and 16 super families in metazoans and fungi, respectively, based on same model structures as plants (Supplementary Tables S9–S10). Consistent with the plant results, metazoans and fungi both showed the worse performance in **ClassII_sub1** than in other models, and most groups (66%; 21/32) in metazoans and (67%; 16/24) in fungi had evaluation scores over 0.7 of PR, SE and *F*1 (Supplementary Tables S9 and S10).

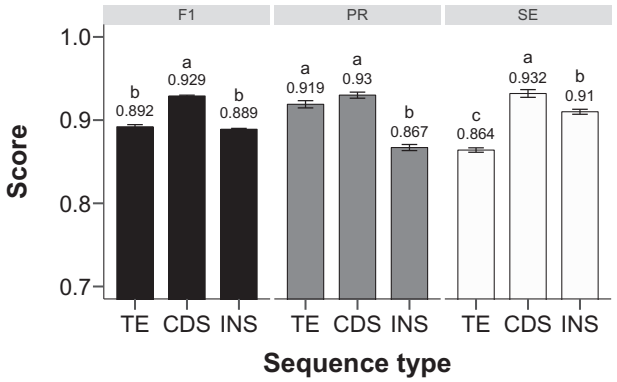

Fig. 4. Comparison of performance for distinguishing non-TE and TE sequences. CDS, coding sequences; INS, intergenic sequences; *F*1, *F*1-score; PR, precision. SE means sensitivity. Different letters (a, b, c) indicate significant differences between scores

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022

4 Discussion

Many studies applied deep-learning models to predict unknown genomic sequences directly using sequence information, instead of predefined features (Eraslan *et al.*, 2019; Park and Kellis, 2015). CNN is an essential model of deep learning, and is suitable for identifying sequence profiles, due to its excellent feature extraction capability on high-dimensional data (Kelley *et al.*, 2016; Zeng *et al.*, 2016). The input vector of CNN is primarily based on sequence-derived features, such as the frequency of *k*-mer occurrence applied in this study and one-hot vector strategy (Aoki and Sakakibara, 2018; Fiannaca *et al.*, 2015; Ghandi *et al.*, 2014; Lee *et al.*, 2011; Nguyen *et al.*, 2016). One apparent advantage of the one-hot vector is to reserve specific position information of each individual nucleotide in sequences. In the case of TE classification, because TEs are different in their sizes, one-hot vector is not directly applicable. Alternatively, the *k*-mer approach can be directly applied to sequences with different sizes and the location of the *k*-mer inside a sequence does not need to be fixed (Eraslan *et al.*, 2019). This method fits TE structure well, since TEs are typically classified into different families based on motif categories or flanking sequence patterns, which can be captured by *k*-mers (Wicker *et al.*, 2007).

We tested different hyper-parameters including the number of network layers, kernel sizes, kernel numbers and *k*-mer sizes. We found these hyper-parameters had small impacts on the model performance except for the *k*-mer sizes. We have compared performance of different *k*-mer sizes, and found that models using $k=7$ performed best. We also tested $k=8$ or larger numbers, but the time it took to train the model was prohibitive, as for the *k*-mer number exponentially increases with *k*-mer length. In **ClassII_sub1** model, we did observe that $k=7$ showed better performance than $k=6$ (Fig. 2). However, PR of **LINE**, **SINE** and **Domain** models did not increase as *k*-mer size increased. **Class**, **ClassI**, **LTR** and **nLTR** models did not show significant improvement between $k=6$ and $k=7$. This observation suggests that the PR may not be sensitive to variations of *k*-mer size in some families but not in others. For example, in the **LINE** model, L1 and I super families could be distinguished since the I superfamily has an additional RNase (RH) domain compared with the L1 superfamily. In contrast, domains of all Class II_sub1 super families are TR, resulting in significant different (*P*-value 0.05) performances among all five *k*-mer sizes (Fig. 2).

DeepTE contains one correction step that utilizes conserved domains to correct false classification (Supplementary Fig. S1 and Table S2), which can be confounded by the presence of nested TEs (Gao *et al.*, 2012). For example, when an nLTR TE is inserted into the body region of an LTR TE, DeepTE will assign this TE to nLTR class since it includes an 'EN' domain. DeepTE is not specifically designed to classify nested TE and the databases we used do not include annotations for nested TEs. Because of this limitation, the correction step is optional when using the DeepTE package. Other published software packages (Gao *et al.*, 2012; Kronmiller and Wise, 2008; Zeng *et al.*, 2017) that are specifically designed to identify nested TEs can be used to annotate these specific types of TEs.

Currently, more than 27 and 17 TE super families are identified, but DeepTE can only classify TEs into 24 and 16 super families in metazoans and fungi (Wicker *et al.*, 2007). The missing families were not considered because there is only a small number of TEs in the databases. In plants, there are only 15 known super families and DeepTE can classify all these TE super families. However, these super families have varied number of TEs, which potentially reduce the classification AC (Barandela *et al.*, 2004). For example, the number of TE/Gypsy (30 368) is 150 times more than the number of TE/P in plants (Supplementary Table S1). To handle this imbalance in training data, we leveraged a tree structured classification process (Fig. 1) by setting eight models that could classify the super families under each order. Among these models, performance of **ClassII_sub1** model was lower than others (Table 1). This could due to that most Class II_sub1 super families are distinguished by sequence dissimilarities of TIR and TSD sizes, rather than motifs in TE body region (Wicker *et al.*, 2007), which could cause difficulties to identify the sequence differences.

5 Conclusion

With the growing availability of reference genomes from non-model species, TE annotation has become a new challenge. In this study, we developed a deep-learning tool called DeepTE to classify unknown TEs based on CNNs, and it outperforms current PASTEC tool. DeepTE contains nine models for different classification purposes, and also include a function to correct false classification based on domain structure. This tool classifies TEs into 15–24 super families according to the sequences from plants, metazoans and fungi. For unknown sequences, DeepTE can distinguish non-TEs and TEs in plant species. A manual is provided for users to apply DeepTE in identification and annotation of TEs in a different genome (Supplementary Note). This tool provides a new method for using deep learning on TE classification.

Funding

This work was supported by USDA Hatch program and translational plant sciences program at Virginia Tech.

Conflict of Interest: none declared.

References

Abdi,H. and Williams,L.J. (2010) Tukey's honestly significant difference (HSD) test. *Encyclopedia of Research Design*. Sage, Thousand Oaks, CA, pp. 1–5.

Abrusán,G. *et al.* (2009) TEclass—a tool for automated classification of unknown eukaryotic transposable elements. *Bioinformatics*, **25**, 1329–1330.

Agarap,A.F. (2018) Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv: 1803.08375*.

Alipanahi,B. *et al.* (2015) Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat. Biotechnol.*, **33**, 831–838.

Aoki,G. and Sakakibara,Y. (2018) Convolutional neural networks for classification of alignments of non-coding RNA sequences. *Bioinformatics*, **34**, i237–i244.

Bao,W. *et al.* (2015) Repbase update, a database of repetitive elements in eukaryotic genomes. *Mob. DNA*, **6**, 11.

Barandela,R. *et al.* (2004) The imbalanced training sample problem: under or over sampling? In: *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. pp. 806–814. Springer ,New York, USA.

Bourque,G. *et al.* (2018) Ten things you should know about transposable elements. *Genome Biol.*, **19**, 199.

Chollet,F. (2015) *Keras Package If Freely Available Online*. https://github.com/fchollet/keras, (September 2019, date last accessed).

Crescente,J.M. *et al.* (2018) MITE Tracker: an accurate approach to identify miniature inverted-repeat transposable elements in large genomes. *BMC Bioinformatics*, **19**, 348.

Ding,J. *et al.* (2010) MiRenSVM: towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features. *BMC Bioinformatics*, **11**, S11.

Eddy,S. (2010) *HMMER3: A New Generation of Sequence Homology Search Software*. http://hmmer.org/, (July 2019, date last accessed).

Ellinghaus,D. *et al.* (2008) LTRharvest, an efficient and flexible software for de novo detection of LTR retrotransposons. *BMC Bioinformatics*, **9**, 18.

Eraslan,G. *et al.* (2019) Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.*, **20**, 389–403.

Feschotte,C. *et al.* (2009) Exploring repetitive DNA landscapes using REPCLASS, a tool that automates the classification of transposable elements in eukaryotic genomes. *Genome Biol. Evol.*, **1**, 205–220.

Fiannaca,A. *et al.* (2018) Deep learning models for bacteria taxonomic classification of metagenomic data. *BMC Bioinformatics*, **19**, 198.

Fiannaca,A. *et al.* (2015) A k-mer-based barcode DNA classification methodology based on spectral representation and a neural gas network. *Artif. Intell. Med.*, **64**, 173–184.

Flutre,T. *et al.* (2011) Considering transposable element diversification in de novo annotation approaches. *PLoS One*, **6**, e16526.

Gao,C. *et al.* (2012) Characterization and functional annotation of nested transposable elements in eukaryotic genomes. *Genomics*, **100**, 222–230.

Ghandi,M. *et al.* (2014) Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput. Biol.*, **10**, e1003711.

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022

Goerner-Potvin,P. and Bourque,G. (2018) Computational tools to unmask transposable elements. *Nat. Rev. Genet.*, **19**, 688–704.
Han,Y. and Wessler,S.R. (2010) MITE-Hunter: a program for discovering miniature inverted-repeat transposable elements from genomic sequences. *Nucleic Acids Res.*, **38**, e199.
Hoede,C. *et al.* (2014) PASTEC: an automatic transposable element classification tool. *PLoS One*, **9**, e91929.
Hu,J. *et al.* (2018) MiteFinderII: a novel tool to identify miniature inverted-repeat transposable elements hidden in eukaryotic genomes. *BMC Med. Genomics*, **11**, 101.
Kamath,U. *et al.* (2014) Effective automated feature construction and selection for classification of biological sequences. *PLoS One*, **9**, e99982.
Kelley,D.R. *et al.* (2016) Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.*, **26**, 990–999.
Kingma,D.P. and Ba,J. (2014) Adam: a method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.
Krizhevsky,A. *et al.* (2012) Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1106–1114.
Kronmiller,B.A. and Wise,R.P. (2008) TEnest: automated chronological annotation and visualization of nested plant transposable elements. *Plant Physiol.*, **146**, 45–59.
Lee,D. *et al.* (2011) Discriminative prediction of mammalian enhancers from DNA sequence. *Genome Res.*, **21**, 2167–2180.
Madden,T. (2013) The BLAST sequence analysis tool. In: McEntyre,J. and Ostell,J. (Eds.) *The NCBI Handbook, National Library of Medicine*, Bethesda, MD (2002), pp. 1–18.
Makałowski,W. (2001) The human genome structure and organization. *Acta Biochim. Pol.*, **48**, 587–598.
McCarthy,E.M. and McDonald,J.F. (2003) LTR_STRUC: a novel search and identification program for LTR retrotransposons. *Bioinformatics*, **19**, 362–367.
Nguyen,N.G. *et al.* (2016) DNA sequence classification by convolutional neural network. *J. Biomed. Sci. Eng.*, **9**, 280–286.
Ou,S. and Jiang,N. (2018) LTR_retriever: a highly accurate and sensitive program for identification of long terminal repeat retrotransposons. *Plant Physiol.*, **176**, 1410–1422.
Park,Y. and Kellis,M. (2015) Deep learning for regulatory genomics. *Nat. Biotechnol.*, **33**, 825–826.
Platt,R.N. *et al.* (2016) Accurate transposable element annotation is vital when analyzing new genome assemblies. *Genome Biol. Evol.*, **8**, 403–410.
Quesneville,H. *et al.* (2005) Combined evidence annotation of transposable elements in genome sequences. *PLoS Comput. Biol.*, **1**, e22.
Rho,M. *et al.* (2007) De novo identification of LTR retrotransposons in eukaryotic genomes. *BMC Genomics*, **8**, 90.
SanMiguel,P. *et al.* (1996) Nested retrotransposons in the intergenic regions of the maize genome. *Science*, **274**, 765–768.
Schmidhuber,J. (2015) Deep learning in neural networks: an overview. *Neural Netw.*, **61**, 85–117.
Schnable,P.S. *et al.* (2009) The B73 maize genome: complexity, diversity, and dynamics. *Science*, **326**, 1112–1115.
Silva,J.C.F. *et al.* (2019) Machine learning approaches and their current application in plant molecular biology: a systematic review. *Plant Sci.*, **284**, 37–47.
Smit,A.F. and Hubley,R. (2008) *RepeatModeler Open-1.0*. http://www.repeatmasker.org/RepeatModeler/, (July 2019, date last accessed).
Spannagl,M. *et al.* (2016) PGSB PlantsDB: updates to the database framework for comparative plant genome research. *Nucleic Acids Res.*, **44**, D1141–D1147.
Steinbiss,S. *et al.* (2009) Fine-grained annotation and classification of de novo predicted LTR retrotransposons. *Nucleic Acids Res.*, **37**, 7002–7013.
Su,W. *et al.* (2019) TIR-Learner, a new ensemble method for TIR transposable element annotation, provides evidence for abundant new transposable elements in the maize genome. *Mol. Plant*, **12**, 447–460.
Wicker,T. *et al.* (2007) A unified classification system for eukaryotic transposable elements. *Nat. Rev. Genet.*, **8**, 973–982.
Xu,Z. and Wang,H. (2007) LTR_FINDER: an efficient tool for the prediction of full-length LTR retrotransposons. *Nucleic Acids Res.*, **35**, W265–W268.
Ye,C. *et al.* (2016) detectMITE: a novel approach to detect miniature inverted repeat transposable elements in genomes. *Sci. Rep.*, **6**, 19688.
Zeng,F.-C. *et al.* (2017) LTRtype, an efficient tool to characterize structurally complex LTR retrotransposons and nested insertions on genomes. *Front. Plant Sci.*, **8**, 402.
Zeng,H. *et al.* (2016) Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*, **32**, i121–i127.
Zhou,J. and Troyanskaya,O.G. (2015) Predicting effects of noncoding variants with deep learning–based sequence model. *Nat. Methods*, **12**, 931–934.

Downloaded from https://academic.oup.com/bioinformatics/article/36/15/4269/5838183 by guest on 21 February 2022