UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE BIOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM GENÉTICA E BIOLOGIA MOLECULAR

ANOTAÇÃO E CLASSIFICAÇÃO DE ELEMENTOS TRANSPONÍVEIS COM O USO DE *DEEP LEARNING*

TIAGO MINUZZI FREIRE DA FONTOURA GOMES

Tese submetida ao Programa de Pós-graduação em Genética e Biologia Molecular da UFRGS como requisito parcial para a obtenção do grau de **Doutor em Genética e Biologia Molecular**.

Orientador: Élgion Lúcio da Silva Loreto

Porto Alegre, Agosto de 2023.

"*A matemática é a arte do perfeito. A física é a arte do ótimo. A biologia, por causa da evolução, é a arte do satisfatório.*"[1]

Sydney Brenner

[1] Trecho extraído do livro "O que é a vida? Compreendendo a biologia em 5 passos", Paul Nurse.

INSTITUIÇÕES E FONTES FINANCIADORAS

Este trabalho foi desenvolvido no Laboratório de Genética e Biologia Molecular (LabDros) na Universidade Federal de Santa Maria. O autor desta tese contou com apoio de uma bolsa de doutorado do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). A parte experimental deste trabalho foi financiada por recursos do LabDros/UFSM e do Programa de Pós-Graduação em Genética e Biologia Molecular (PPGBM) da UFRGS.

AGRADECIMENTOS

A minha mãe, exemplo de pessoa forte, dedicada, cuidadosa e amorosa, a qual sem a confiança, apoio e suporte eu não teria conseguido chegar até aqui. As palavras tornam-se insuficientes para expressar toda a gratidão e o amor que sinto por ela.

A minha namorada, companheira e amiga Vanessa que me acompanha desde o começo desta jornada, sendo também porto seguro inabalável, que sempre me motivou e me ajudou a levantar em todos os momentos difíceis deste caminho. Amo-te.

Aos meus queridos 'pets', os que já se foram e os que ainda estão conosco, por ajudarem a ver a vida de forma simples e leve, proporcionando momentos alegres à nossa família.

Ao professor Élgion por desafiar com este projeto e por todo o ensinamento científico proporcionado nesses anos todos de convivência, e a todo mundo do LabDros pela amizade, especialmente a Mariana e a Daniela, colegas, amigas e parceiras de publicações. Também, a todos os amigos e amigas feita(o)s "em laboratório" ao longo dos anos como a Raquel, Daniel, Pedro, Nader, Sinara, Francine, Zé, Marcos, Mônica entre outros.

A minha amiga Valéria, pessoa divertida, inteligente, exemplo de pesquisadora apaixonada pela ciência, que me serve de espelho para buscar melhorar como cientista e me ajudou a seguir na ciência com nossas conversas.

Ao Elmo, que apesar da distância, sempre foi de muita ajuda durante o processo do doutorado e vejo como o coração do PPGBM, o qual todos os discentes e docentes tenho certeza também o admiram.

Por fim, a todo mundo que ao longo do caminho contribuiu de uma forma ou de outra para eu poder chegar até aqui. Afinal, ninguém é uma ilha. O trajeto seria muito mais tortuoso sem os aprendizados e suporte obtidos.

SUMÁRIO

LISTA DE ABREVIATURAS

Adagrad: *adaptative gradient descent*.

CNN: *convolutional neural networks*.

DL: *deep learning*.

ERV: *endogenous retrovirus*.

HS: *host shift*.

IA: inteligência artificial.

LINE: *long interspersed nuclear element.*

LISP: acrônimo de *list processing*.

LTR: *long terminal repeats*.

MAE: *mean absolute error*.

MLP: *multilayer perceptron*.

MSE: *mean squared error*.

piRNA: *Piwi-interacting* RNA.

ReLU: *rectified linear unit*.

SINE: *short interspersed nuclear element.*

siRNA: *small interference* RNA.

SVM: *support vector machine*.

TEs: elementos transponíveis. Do inglês, *transposable elements*.

RESUMO

Os elementos transponíveis (TEs) são sequências de DNA capazes de se transporem dentro de um genoma hospedeiro e desempenham vários papéis na regulação dos genes, no envelhecimento, no desenvolvimento de certos tipos de câncer, na especiação e no desenvolvimento do sistema imunológico, entre outros. A identificação e classificação dos TEs nos genomas constituem um desafio devido à sua natureza repetitiva e diversificada. Embora se aplique várias técnicas para a anotação de TEs, o ressurgimento de *deep learning* (DL) trouxe novas possibilidades dentro das ciências ômicas com esta finalidade. As redes neurais convolucionais (CNN) têm sido aplicadas com sucesso em vários domínios, incluindo a classificação de imagens, o processamento de linguagem natural e na genômica. No entanto, faltam ferramentas baseadas em DL que possam efetuar a identificação e classificação de TEs de ponta a ponta. Nesta tese, apresentamos o HamleTE, uma ferramenta baseada em DL que utiliza um *workflow* para anotar e classificar TEs em genomas. HamleTE oferece os modos de anotação e classificação, proporcionando flexibilidade para diferentes casos de uso. A ferramenta emprega CNNs para extração de características, seguida por camadas totalmente conectadas para aprender as associações entre dados e rótulos para categorização precisa. Ao contrário das ferramentas existentes, HamleTE integra etapas de extração de sequências repetitivas e de remoção de redundância, assegurando uma anotação TE robusta. Para avaliar o desempenho do HamleTE, comparamo-lo com outros programas de classificação de TE. Os resultados demonstraram que, em relação aos outros programas, HamleTE alcançou um desempenho comparável ou superior em termos de identificação correta de TEs, precisão, especificidade, acurácia, sensibilidade e F1-score. Além disso, o modo de anotação do HamleTE gerou bibliotecas de TEs que refletem com precisão a distribuição de TEs em diferentes espécies, superando os programas de anotação existentes em termos de representação e cobertura. Sua fácil instalação e utilização, bem como eficiente uso de recursos computacionais, tornam HamleTE acessível tanto a especialistas em bioinformática como a não especialistas. Para resolver os desafios da classificação de TE, HamleTE

emprega um *workflow* hierárquico com vários modelos de classificação. Esta abordagem reduz a complexidade e a variação em cada etapa, atenuando as dificuldades associadas à aprendizagem e à categorização. Além disso, o HamleTE utiliza *embedding vectors* para representar sequências de DNA, capturando as relações contextuais e a semântica da informação genética. Esta abordagem melhora a capacidade do modelo para extrair características e aumenta a precisão da classificação. Em conclusão, HamleTE preenche a lacuna nas ferramentas de anotação e classificação de TE baseadas em DL. Ele fornece um *workflow* abrangente e eficiente para a análise de TEs, fornecendo resultados precisos e possibilitando opções de refinamento dos resultados. Ao tirar partido do poder da DL, HamleTE permite aos pesquisadores explorar a paisagem repetitiva e diversificada dos TEs nos genomas eucarióticos, facilitando uma exploração dos seus papéis funcionais e evolutivos.

Palavras-chave: anotação, bioinformática, *deep learning*, elementos transponíveis.

ABSTRACT

Transposable elements (TEs) are DNA sequences capable of transposing within a host genome, and they play various roles in gene regulation, aging, cancer, speciation, and immune system development, among other processes. Accurate identification and classification of TEs in genomes are challenging due to their repetitive and diverse nature. While several techniques have been developed for TE annotation, the recent re-emergence of deep learning has provided new opportunities for omics sciences. Convolutional neural networks (CNNs) have been successfully applied in various domains, including image classification, natural language processing, and now, genomics. However, there is a lack of deep learning-based tools that can perform end-to-end TE identification and classification. In this thesis, we present HamleTE, a deep learning-powered tool that utilizes a workflow to annotate and classify TEs in genomes. HamleTE offers both annotation and classification modes, providing flexibility for different use cases. The tool employs CNNs for feature extraction, followed by fully-connected layers to learn the associations between data and labels for accurate categorization. Unlike existing tools, HamleTE integrates repeat extraction and redundancy removal steps, ensuring robust TE annotation. To evaluate HamleTE's performance, we compared it with other TE classification programs. The results demonstrated that HamleTE achieved comparable or superior performance in terms of correct TE identification, precision, specificity, accuracy, recall, and F1-score. Furthermore, HamleTE's annotation mode generated TE libraries that accurately reflected the distribution of TEs in different species, outperforming existing annotation programs in terms of representation and coverage. The tool's user-friendly installation and usage, as well as its efficient resource utilization, make it accessible to both bioinformatics experts and non-specialists. To address the challenges of TE classification, HamleTE employs a hierarchical workflow with multiple classification models. This approach reduces complexity and variance at each step, mitigating the difficulties associated with learning and categorization. Furthermore, HamleTE utilizes embedding vectors to represent DNA sequences, capturing the contextual relationships and semantic of the genetic information.

This approach improves the model's ability to extract features and enhances classification accuracy. In conclusion, HamleTE fills the gap in deep learning-based TE annotation and classification tools. It provides a comprehensive and efficient workflow for TE analysis, delivering accurate results and allowing options for curating the results. By leveraging the power of deep learning, HamleTE enables researchers to explore the repetitive and diverse landscape of TEs in eukaryotic genomes, facilitating the exploration of their functional and evolutionary roles.

Keywords: annotation, bioinformatics, deep learning, transposable elements.

CAPÍTULO 1

INTRODUÇÃO GERAL

1. REDES NEURAIS ARTIFICIAIS

Machine learning ou aprendizagem de máquina é um campo das ciências da computação, pertencente ao grande campo da inteligência artificial (IA), que busca desenvolver algoritmos que possam aprender características a partir de um conjunto de dados (Shinde e Shah 2018). *Deep learning*, ou aprendizado de máquina profundo, é um subcampo dentro da área de aprendizagem de máquina que utiliza múltiplas camadas de redes neurais artificiais, visando a emular o comportamento do cérebro humano, para aprender padrões complexos e estabelecer relações entre dados (LeCun e cols. 2015). Tem como aplicações, por exemplo, análise de mercado financeiro (Yekrangi e Nikolov 2023; Blasco e cols. 2023), reconhecimento de imagens (Rawat e Wang 2017; Luo e cols. 2018), processamento de linguagem natural (Wang e Gang 2018; Han e cols. 2020), inclusive, com aplicações dentro das diferentes áreas das ciências naturais e da saúde (Cao e cols. 2018; Ching e cols. 2018; Oubounyt e cols. 2019; Martorell-Marugán e cols. 2019).

1.1 Breve histórico da pesquisa em IA

Podemos traçar o começo da pesquisa na área da aprendizagem de máquina a partir dos anos 40 do século XX. Os neurofisiologista Warren McCulloch e o matemático Walter Pitts, no ano de 1943, publicaram o artigo "*A Logical Calculus of Ideas Immanent in Nervous Activity*" apresentando um modelo de neurônio artificial baseado no funcionamento dos neurônios humanos. Este neurônio artificial, por meios algorítmicos e matemáticos, simularia o processamento da informação das conexões neurais do cérebro humano aplicando uma lógica de limiar, no qual um neurônio artificial seria ativado caso ultrapassasse um certo limiar de ativação após o processamento de sinais de entrada (*inputs*) recebidos (McCulloch e Pitts 1943; Cowan 1990). Em 1957 o psicólogo Frank Rosenblatt desenvolveu o algoritmo do *Perceptron* (Figura 1), baseando-se no neurônio artificial de McCulloch-Pitts, com o objetivo de melhorar as predições computacionais ao aprender padrões e ajustar os parâmetros do modelo até atingir valores ótimos capazes de classificar corretamente os dados (Rosenblatt 1958; Seising 2018).

O *Perceptron* foi um dos primeiros algoritmos de redes neurais artificiais e foi inicialmente projetado para tarefas como o reconhecimento de dígitos escritos à mão (Kussul e cols. 2001). Ele produz uma saída binária (*i.e.*, 0 ou 1, verdadeiro ou falso) a partir de diversos sinais de entrada os quais são multiplicados por um dado peso, e ao somatório destas multiplicações adiciona-se um viés (*bias*, em inglês) (Rosenblatt 1962; Kanal 2003). Os pesos são os parâmetros que definem o quão relevante é um sinal de entrada, ou seja, qual a influência deste dentro do conjunto de dados. O viés pode ser entendido como o fator que determina o quão fácil um neurônio artificial pode ser ativado. Um viés muito positivo facilita a ativação, enquanto um muito negativo dificulta a ativação. Se o resultado for maior que um determinado limiar o neurônio artificial é ativado (Cowan 1990; Wang e cols. 2018; El-Amir e Hamdy 2020). Ambos peso e viés são parâmetros auto-ajustáveis do modelo, isto é, eles são aprendidos durante o treinamento. Essa capacidade de aprender os parâmetros é o que diferencia o *Perceptron* do neurônio de McCulloch-Pitts e levou ao seu sucesso.

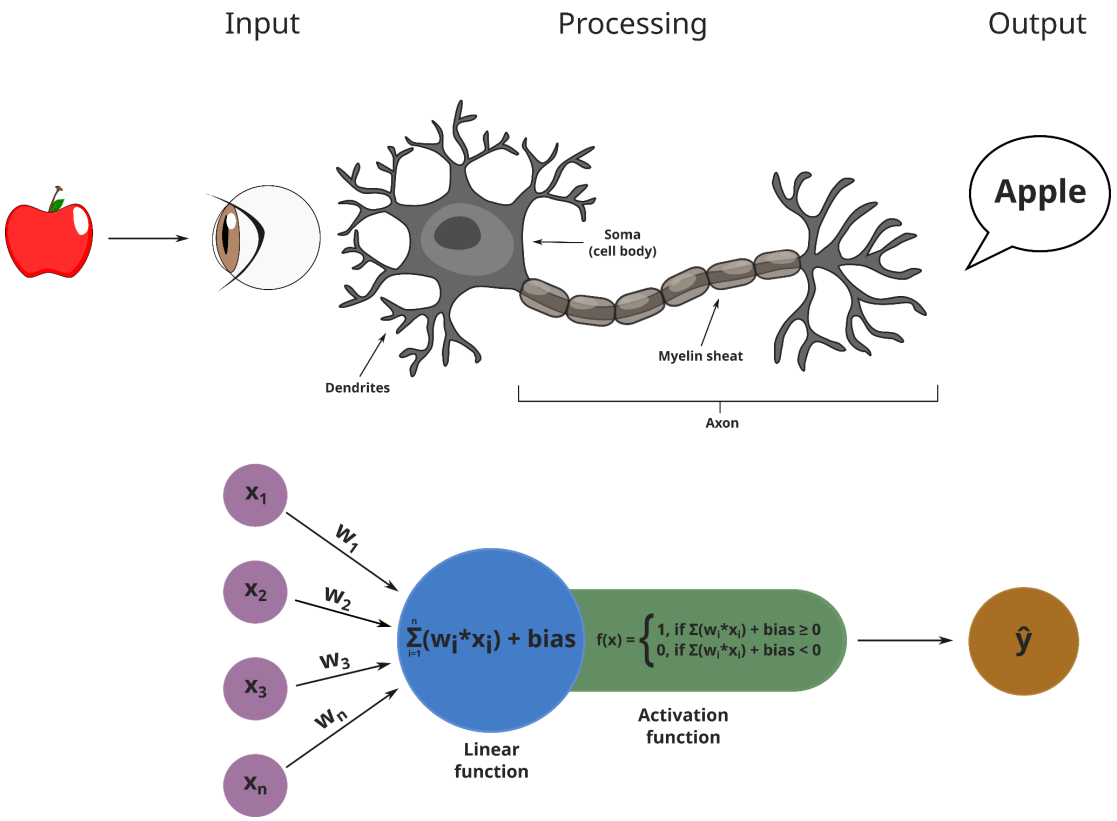


Figura 1. Representação esquemática comparando um neurônio humano e o processamento da informação com o *perceptron* de Rosenblatt. Fonte: autores.

Apesar do êxito inicial do método, o *perceptron* em sua modelagem original apresentava limitações significativas, pois era capaz de classificar apenas dados linearmente separáveis. A insuficiência de sua aplicação em tarefas mais complexas associada a baixa velocidade de processamento e baixa memória dos computadores na época, levou ao chamado primeiro inverno da IA (Toosi e cols. 2021) - década de 1970 até início dos anos 1980 - onde muitos projetos foram terminados pelo corte de financiamento ocorrido pelo insucesso das pesquisas na área como, por exemplo, no campo de *machine translation*, um ramo da linguística computacional com o objetivo de realizar tradução de um idioma para o outro por meios computacionais (Hendler 2008; Floridi 2020). Um ponto de virada se deu com a implementação da retropropagação na pesquisa de IA, que apesar de ter sido pensada no fim dos anos 60, ganhou popularidade após a publicação de Rumelhart e colaboradores em 1986. A retropropagação é considerada como um dos pontos mais importantes do campo, tendo permitido o treinamento de modelos com uma eficiência não antes vista. Foi possível diminuir tempos de treinamento de modelos que outrora duravam dias ou mais para questão de horas (Wythoff 1993; Rojas 1996).

Um segundo inverno da IA ocorreu do fim dos anos 1980 até meados dos anos 1990 devido ao colapso das máquinas LISP, as quais eram o referencial dentre os pesquisadores e desenvolvedores de IA na época, e eram extremamente caras comparadas aos computadores pessoais que estavam em ascensão na época (Toosi e cols. 2021). A volta do desenvolvimento na área, no que se pode chamar de primavera da IA, se deu a partir do fim dos anos 1990 com eventos como a vitória do computador *Deep Blue* sobre o campeão de xadrez Gary Kasparov (Hsu 1999; Campbell e cols. 2002) e o desenvolvimento de assistentes virtuais no fim da primeira década dos anos 2000. A partir desta época, com publicações de Hinton e Salakhutdinov, o termo *deep learning* começou a se popularizar. Naqueles trabalhos, os autores demonstraram como uma rede neural de várias camadas poderia ser treinada uma camada de cada vez obtendo um melhor aprendizado (Hinton e Salakhutdinov 2006; Salakhutdinov e Hinton 2007).

1.2 Introdução às redes neurais artificiais

Os diferentes algoritmos de aprendizagem de máquina podem ser baseados nos paradigmas de aprendizagem conhecidos como aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço (Alom e cols. 2018; Dong e cols. 2021; Sharma e cols. 2021). No aprendizado supervisionado, o algoritmo é alimentado por dados que são previamente categorizados para que o modelo aprenda a relação entre os dados de entrada e a categoria a qual este dado pertence. Algoritmos que utilizam aprendizagem supervisionada são muito usados para criar modelos de classificação ou regressão - a predição de valores baseados em dados contínuos. Exemplos de aprendizado supervisionado são *random forest*, *support vector machine* (SVM) e algoritmos baseados em redes neurais artificiais (Caruana e Niculescu-Mizil 2006; Saravanan e Sujatha 2018). O aprendizado não-supervisionado diz respeito a algoritmos que aprendem a estabelecer relação entre os dados sem uma prévia categorização destes. Algoritmos de aprendizado não-supervisionado são usados para, por exemplo, modelos de agrupamento ou de sistemas de recomendação, utilizando de algoritmos como *k-means* e *hierarchical clustering* (Celebi e Aydin 2016; Sinaga e Yang 2020). O método de aprendizado por reforço é mais usado em robótica e criação de jogos, casos nos quais o modelo deve aprender uma sequência de eventos por interação com o ambiente por meio de tentativa e erro (Kaelbling e cols. 1996; Li 2018; François-Lavet e cols. 2018).

Algoritmos baseados em redes neurais artificiais estão sendo cada vez mais utilizados devido a sua grande capacidade de fazer predições a partir de uma grande quantidade de dados com alta complexidade, superando outros métodos de aprendizagem de máquina como SVM. Dentre os algoritmos mais basilares ou em voga estão o perceptron multicamadas (MLP, do inglês, *multi-layer perceptron*) e redes neurais convolucionais (CNN, do inglês *convolutional neural networks*). Pode-se considerar o MLP como o exemplo base de uma rede neural artificial moderna (Murtagh 1991; Li e cols. 2023). A arquitetura básica é composta por uma camada de entrada, uma ou mais camadas intermediárias conhecidas como camadas escondidas e, por fim, uma camada de saída (Figura 2). Um MLP com duas ou mais camadas escondidas

pode ser caracterizado como de aprendizado profundo, o *deep learning*. A camada de entrada diz respeito aos dados a serem aprendidos e a camada de saída às predições baseadas no aprendizado (Schmidhuber 2015; Jakhar e Kaur 2020). As camadas escondidas são compostas por nodos (ou neurônios) empilhados em cada camada, sendo os responsáveis pelo armazenamento dos parâmetros aprendidos pela rede neural. Todos os nodos de uma camada estão conectados a todos os nodos de uma próxima camada, originando-se disto o nome de camadas densamente conectadas (Somuncuoğlu e cols. 2020; Uzair e Jamil 2020). Os parâmetros aprendidos são atualizados durante um dado número de etapas para valores que melhor representem os dados vistos até que ocorra o aprendizado adequado destes. A atualização dos parâmetros se dá pela redução da perda com o uso de funções de otimização como o gradiente descendente.

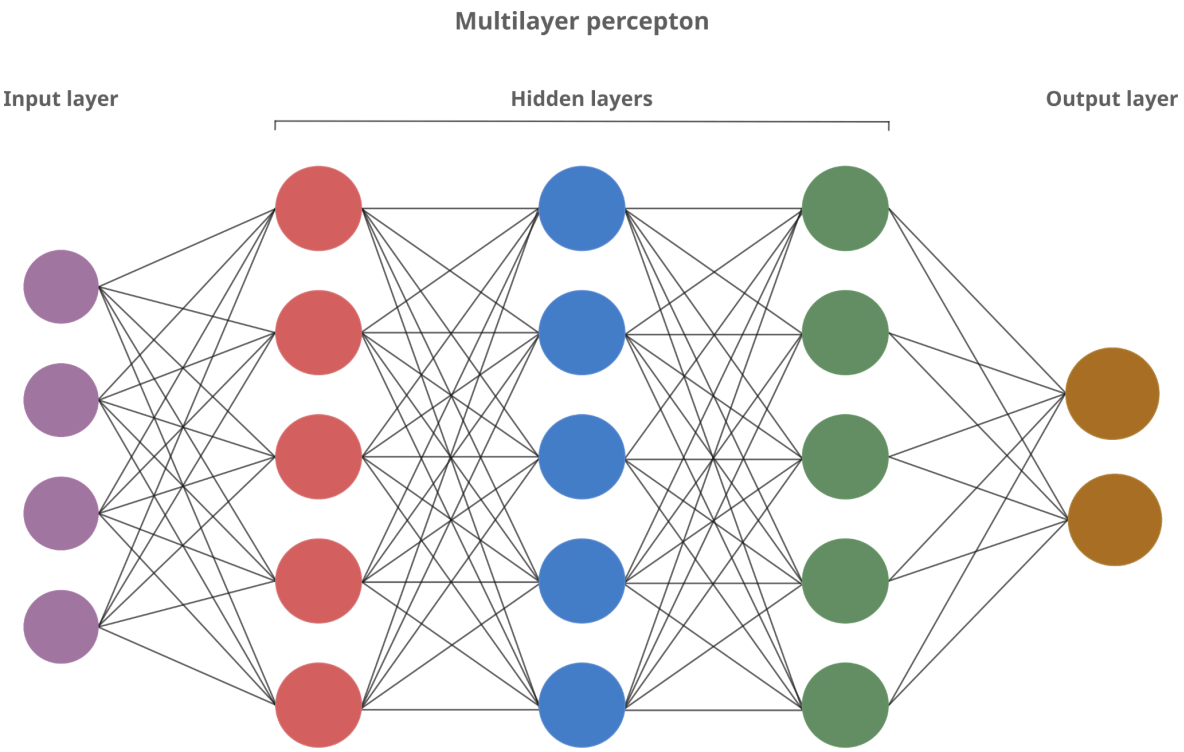


Figura 2. Representação de um *perceptron* multicamadas. Neste exemplo temos a camada de entrada (*input layer*), três camadas escondidas (*hidden layers*) e a camada de saída (*output layer*). Fonte: autores.

A perda (ou custo) do aprendizado é a medida da diferença entre o valor real de um dado e o valor predito, calculada por uma função de perda como MAE,

MSE, *binary cross-entropy* e *categorical cross-entropy*. As funções de *cross-entropy* baseiam-se no cálculo comparativo da soma do produto da distribuição verdadeira e do logaritmo da distribuição prevista pelo modelo, sendo a *binary cross-entropy* utilizada para classificações binárias, como sugere o nome, enquanto a *categorical cross-entropy* é usada para classificações com diversas classes (de Boer e cols. 2005; Gordon-Rodriguez e cols. 2020). Otimizadores como o gradiente descendente possuem como objetivo encontrar um mínimo global para minimizar o valor da função de perda (Schmidt e cols. 2021). Outros otimizadores populares são gradiente descendente estocástico, Adagrad e Adam (Okewu e cols. 2019; Choi e cols. 2020). A atualização dos parâmetros com um otimizador é calculada pela técnica da retropropagação pelo uso da regra de cadeia, que é o método para se calcular a derivada de uma função composta. A retropropagação é chamada desta forma, pois ela calcula, por exemplo, o gradiente descendente a partir da última camada em direção às camadas iniciais para atualizar os valores dos parâmetros aprendidos pela rede neural (Wythoff 1993; Baldi e cols. 2018; Lillicrap e cols. 2020). Os principais parâmetros de uma rede neural são os pesos e o viés. Os pesos são os responsáveis por estabelecer a força das conexões entre os nodos, informando a importância de uma conexão em relação ao dado de saída das categorias a serem preditas. O viés é a constante adicionada ao produto dos pesos com os valores dos dados de entrada, e causa o deslocamento deste valor em um plano, auxiliando a função de ativação a estabelecer a saída (Denil e cols. 2013; Suk 2017). Em outras palavras, os parâmetros, como o peso e o viés são as informações, aprendidas pela rede neural durante o treinamento a fim de estabelecer as conexões necessárias para atingir o resultado.

Outro conceito importante para a construção de uma rede neural é o de hiperparâmetros. Estes, diferentemente dos parâmetros que são a parte aprendida, são os valores configuráveis durante a criação da arquitetura do modelo da rede neural, realizando-se antes do treinamento em si. Os hiperparâmetros ditam o quão eficiente será o aprendizado da rede, necessitando de um ajuste fino dos seus valores a fim de atingir um resultado ótimo de aprendizado. Os hiperparâmetros principais de uma arquitetura de redes neurais

são a taxa de aprendizado, a função de ativação, o número de épocas de treinamento, o número de camadas da rede e o número de nodos em cada camada (Young e cols. 2015; Shankar e cols. 2020). A taxa de aprendizagem é o valor que controla a velocidade do modelo em se adaptar aos dados em questão. Está intimamente relacionada com o gradiente descendente, pois a velocidade dos passos deste na tentativa de encontrar um mínimo global para diminuição do erro é dada pela taxa de aprendizagem. Um modelo com uma taxa de aprendizagem muito baixa pode ser incapaz de aprender e atingir os valores ótimos na descida de gradiente, ficando preso em mínimos locais. Taxas muito altas podem induzir o modelo a aprender de forma muito rápida obtendo resultados muito distantes do ideal (Chandra e Sharma 2016; Johny e Madhusoodanan 2021). A função de ativação introduz a não-linearidade necessária ao *output* de cada neurônio quando se trabalha com dados de alta complexidade, ou seja, faz a transformação dos dados de entrada para o *output* do neurônio. A função de ativação pode ser desde uma simples *step function* como do primeiro perceptron desenvolvido ou outras bastante usadas atualmente como *tahn*, ReLU (do inglês, *rectified linear unit*) e sigmóide. O número de épocas diz respeito a quantos ciclos de treinamento serão realizados pela rede neural até se atingir uma performance adequada. Em termos simples, é o número de vezes que a rede precisa “estudar” os dados até aprender.

O número de camadas e o número de nodos em cada camada, associados aos hiperparâmetros descritos anteriormente, tem relação direta com a capacidade de aprendizagem e generalização do modelo. Por generalização, entende-se a capacidade do modelo de ser aplicado em dados do mundo real não vistos durante o treinamento com a mesma eficiência obtida no treinamento. Dois aspectos fundamentais relacionados a uma limitada capacidade de generalização são o *overfitting* e *underfitting*. O *overfitting* é o ajuste demasiado realizado pelo modelo em relação aos dados de treinamento, podendo ser visualizado por métricas de acurácia bastante altas durante o treinamento, mas com resultados parcos ao se testar com dados não vistos durante o treino (Bejani e Ghatee 2021; Ghojogh e Crowley 2023). Pode ocorrer em modelos treinados com dados insuficientes, modelos com arquiteturas muito complexas em relação aos dados,

assim como dados como em caso de dados com muito ruído, levando o modelo a basicamente os “decorar” ao invés de aprender seus padrões gerais. Formas de contornar este problema são a reestruturação da arquitetura do modelo a fim de reduzir a sua complexidade, o aumento do tamanho do dataset de treinamento e o uso de técnicas de regularização como o *dropout*, técnica que desativa aleatoriamente uma porção definida de neurônios da arquitetura em cada fase do treinamento, forçando-o a não se apoiar em porções muito específicas dos dados (Ashiquzzaman e cols. 2018; Gavrilov e cols. 2018; Rice e cols. 2020). O *underfitting* é a falta de aprendizado durante o treinamento e pode ser visualizado de forma mais clara, pois durante o treino as métricas são baixas e de avanço lento (Gavrilov e cols. 2018; Liu e cols. 2023). Modelos muito simples, falta de treinamento, dados muito complexos ou não representativos e dados insuficientes podem causar este fenômeno. Soluciona-se ao aumentar o número de neurônios e/ou camadas na arquitetura do modelo, usando-se dados mais representativos ou aumentando o tempo de treino (Kolluri e cols. 2020; Li e cols. 2021).

As CNN são uma classe de redes neurais de aprendizagem profunda que ganharam visibilidade por seu desempenho na classificação e reconhecimento de imagens, reconhecimento de fala, detecção de objetos e análise de dados de séries temporais (O’Shea e Nash 2015; Wang e Gang 2018; Wang e Huang 2023). A arquitetura padrão de uma CNN é composta por uma ou mais camadas convolucionais, camadas de *pooling* e camadas densamente conectadas, além das camadas de entrada e saída. O nome advém do uso da operação matemática de convolução a qual gera uma terceira função a partir da soma dos produtos de duas outras funções nos pontos onde elas se sobrepõem. Em termos práticos, esta operação é usada na CNN ao se realizar a extração de características dos dados com o uso de um ou mais filtros (também chamados *kernels*) deslizantes que geram um mapa de características (Figura 3). Aplica-se às camadas convolucionais uma função de ativação e então uma camada de *pooling*. Esta camada tem por objetivo reduzir a dimensionalidade do mapa de características e o número de parâmetros da rede ao extrair os pontos identificados como os mais relevantes (Alom e cols. 2018; Tao e cols. 2022; Wang e Huang 2023). O *pooling* pode ser realizado mais comumente pela operação de *max pooling* (Mehedi

Shamrat e cols. 2021), a qual extrai o maior valor de uma seção do mapa de características de acordo com a dimensão da camada de *pooling*, ou de *average pooling*, o qual extrai a média dos valores da seção (Figura 3).

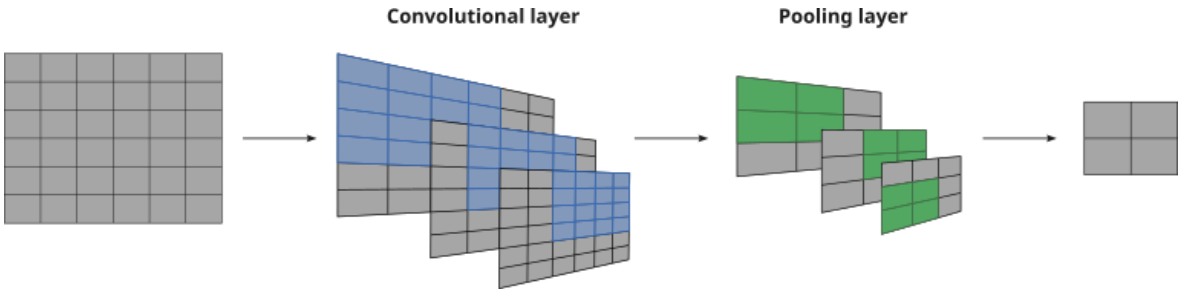


Figura 3. Etapas de extração de características de uma rede neural convolucional com o uso das camadas convolucional e de *pooling*. Em azul temos o filtro (*kernel*) da camada de convolução e em verde vemos o filtro de *pooling* que pode extrair o valor médio ou máximo da seção analisada de acordo com o tipo de *pooling* utilizado. Fonte: autores.

Para se treinar uma rede neural, além de se estabelecer a arquitetura do modelo e o algoritmo usado, o conjunto de dados deve ser representativo do problema a ser resolvido, recomendando-se haver um número igual de elementos para cada categoria a fim de evitar vieses de treinamento em favor de uma das categorias a serem aprendidas (Shahinfar e cols. 2020). Além disso, sugere-se a divisão do conjunto de dados em dados de treino e de validação, em uma proporção de, por exemplo, 80-20 ou 90-10, respectivamente (Zou e cols. 2019). A maior porcentagem dos dados é usada para realizar o treinamento do modelo, enquanto a menor parte para avaliar o modelo com dados não vistos durante o treinamento. Desta forma, pode-se verificar a ocorrência ou não dos fenômenos de *overfitting* e *underfitting*, assim como a performance do modelo de forma geral. Deve-se atentar a presença da proporcionalidade de cada categoria nos conjuntos de dados de treino e validação, isto é, todas as categorias devem ter proporções iguais entre os conjuntos de dados de treino e validação (Chen e cols. 2017). Apesar destas recomendações, nem sempre é possível a coleta de dados em número igual e suficiente para cada categoria, levando a conjuntos de dados desbalanceados (Johnson e Khoshgoftaar 2019a; Saini e Susan 2022).

Para contornar estes problemas se pode utilizar de estratégias como *oversampling* e *undersampling* (ou *downsampling*) (Johnson e Khoshgoftaar 2019b; Lashgari e cols. 2020; García-Ordás e cols. 2021). O primeiro consiste em aumentar de forma sintética o número de amostras de categorias com menos elementos até se igualarem os números de amostras de todas as categorias (Yedida e Menzies 2022). No *undersampling* ocorre o oposto, diminui-se o número de elementos da categoria com mais elementos até que se iguale com a de menor número (Shahinfar e cols. 2020). O uso de *oversampling* pode levar o modelo ao overfitting para a categoria em questão, além de aumentar o tempo de treinamento devido ao aumento do número de dados, e, dependendo do tipo de dados, pode não ser possível realizá-lo de forma a evitar este viés (Roy e cols. 2019; Tarekegn e cols. 2021). No caso do *undersampling*, os dados da categoria submetida a este método podem não ser representativos da categoria, enviesando o aprendizado ao se descartar os dados que poderiam ser mais relevantes (Liu e cols. 2022). Outra forma de se trabalhar com dados desbalanceados é através da atribuição de pesos de classe previamente ao treinamento. Neste caso, o peso inicial de uma classe é inversamente proporcional ao número de elementos nela contidos, assim, durante o treinamento, há uma penalidade maior para os erros em classes com menor número de elementos (Krawczyk e cols. 2014; Cui e cols. 2019; Gao e cols. 2020). Ao se trabalhar com dados desbalanceados, pode-se também mesclar as diferentes estratégias como medida de se obter melhores resultados (Hernandez e cols. 2013).

2. ELEMENTOS DE TRANSPOSIÇÃO

Elementos de transposição (TEs) são segmentos de DNA capazes de se inserir em uma nova posição no genoma. Foram descobertos por Barbara McClintock na década de 1940 ao observar a ocorrência de diferentes fenótipos de grãos em espigas de milho. Ao investigar o fenômeno, ela descobriu a presença de um locus no cromossomo 9 o qual denominou de *Ds* (do inglês *dissociation*), responsável pela variegação do milho. Após a descrição deste fenômeno ocasionado pela inserção do elemento *Ds*, Barbara McClintock

descobriu que este elemento não era capaz de se mobilizar sozinho, necessitando da ação de um outro elemento o qual foi nomeado *Ac* (do inglês, *activator*) (McClintock 1947; McClintock 1956). Os achados de McClintock foram encarados inicialmente com muitas restrições e ceticismo por grande parte da comunidade científica da época, visto que a ideia de segmentos de DNA com capacidade móvel ia contra o pensamento corrente encabeçado por cientistas como Thomas H. Morgan de que os genes eram estruturas estáticas de posição definida nos genomas, relegando a descoberta de McClintock a um *status* marginal à ciência corrente. Foi apenas a partir da década de 1970 com a descoberta de elementos transponíveis em bactérias, em *Saccharomyces cerevisiae* e *Drosophila melanogaster*, por exemplo, que a ciência pode se fazer agente da justiça para recompensar McClintock por sua enorme contribuição à genética, laureando-a como a primeira mulher a ganhar um Nobel não compartilhado de fisiologia ou medicina em 1983, 35 anos após sua descoberta (Biémont 2010; Ravindran 2012).

Outrora adjetivados como DNA “lixo”, DNA egoísta e atribuídos à condição de elementos moleculares parasíticos (Ohno 1972; Doolittle e Brunet 2017), os TEs estão associados a uma profusão de eventos genéticos e adaptativos como especiação (Warren e cols. 2015; Serrato-Capuchina e Matute 2018), diversificação do sistema imune (Broecker e Moelling 2019; Ivancevic e Chuong 2020), gestação interna em mamíferos placentários (Sotero-Caio e cols. 2017), coloração em plantas e animais (Hirsch e Springer 2017; Galbraith e Hayward 2023), e perda da cauda em grandes primatas (Xia e cols. 2021; Hayward e Gilbert 2022), para citar alguns. Os TEs são encontrados em virtualmente todas as espécies eucarióticas já estudadas a nível genômico, contribuindo com grande porção do genoma destas (Bourque e cols. 2018; Wells e Feschotte 2020). O próprio genoma da espécie que levou McClintock à descoberta dos TEs, *Zea mays*, o milho, possui, segundo estimativas, até 85 % do seu genoma composto por TEs (Stitzer e cols. 2021). Outras espécies notáveis por grande porção de TEs no genoma são *Mus musculus*, com aproximadamente 40%, *Hordeum vulgare*, com 55%, *Drosophila melanogaster* até 20% e *Homo sapiens*, 45-50% (Kidwell 2002; Canapa e cols. 2016). No caso do genoma humano, há uma

multitude de resíduos de sequências associados ao elemento *Alu* inseridos ao longo de milhões de anos de evolução e que também podem ser encontrados em outros grandes primatas (Batzer e Deininger 2002; Britten 2010). A inserção destes elementos em certas regiões do genoma está associada à instabilidade genômica durante o envelhecimento e inclusive à doenças como hemofilia, neurofibromatose e câncer de mama (Hedges e Deininger 2007; Andrenacci e cols. 2020).

As inserções de TEs em sua maioria promovem efeitos neutros ou deletérios. Neste último caso, é esperada a eliminação dos elementos em uma população por efeitos de seleção purificadora (Oggenfuss e cols. 2021; Doolittle 2022). A mobilização de TEs causando efeitos deletérios geralmente está associada à ideia de um agente estressor causador de instabilidade. É notório o efeito de certos tipos de estresse na mobilização dos elementos, como, por exemplo, a mobilização de elementos *mariner* em *Drosophila simulans* por estresse térmico (Cancian e cols. 2022), o aumento da expressão de elementos LTR por estresse químico em drosófilas tratadas com cisplatina (Mombach e cols. 2022b) e casos de estresse biótico levando a mobilização de elementos *hAT* em plantas (Deneweth e cols. 2022). Apesar de o termo estresse ser comumente associado à ideia da mobilização de TEs, é importante ressaltar que nem todo tipo de estresse causa mobilização, visto que esta relação estresse-mobilização é complexa e possui muitas variáveis (Mombach e cols. 2022a). Os efeitos danosos dos TEs são combatidos por mecanismos de controle, como a metilação, o silenciamento de TEs por piRNAs e siRNAs (Burns 2017; Nakamura e cols. 2019), e biotinilação de histonas (Zempleni e cols. 2009).

Os cenários quais os TEs estão inseridos podem ser benéficos para a adaptação do organismo em questão, fixando-os na população. O próprio silenciamento de um TE pode levar a cooptação do elemento para realizar novas funções em um genoma. Há famílias de *ERVs* em mamíferos cooptadas como sítios de regulação da cromatina, ou o caso do gene *Arc*, com papel no armazenamento da memória e plasticidade do córtex visual, derivado do gene *gag* de retrotransposons (Hayward and Gilbert 2022). Outro exemplo da influência de TEs evolutivamente é a produção de amilase pelas glândulas salivares em

primatas, onde já foi demonstrada ter ocorrido a exaptação de sequências derivadas do elemento *HERV-E* como promotores, possivelmente auxiliando na melhora da digestão de amido, aumentando o *fitness* deste grupo (Bourque e cols. 2018). Evidências sugerem o papel dos TEs como agentes de plasticidade genômica, conferindo rápida adaptabilidade de organismos ao enfrentar novos desafios ambientais, a exemplo de formigas da espécie *Cardiocondyla obscurior*, e do que ocorre na evolução de patógenos os auxiliando na "corrida armamentista" hospedeiro-patógeno, em um conceito conhecido como genoma de duas-velocidades, onde uma parte do genoma composta por maior porção de elementos repetitivos evolui em maior velocidade do que a parte com menor porção destes elementos (Schrader e Schmitz 2019). De forma geral, os TEs contribuem de forma significativa com a evolução do tamanho dos genomas de invertebrados (Petersen e cols. 2019) e vertebrados (Sotero-Caio e cols. 2017; Shao e cols. 2019).

2.1 Classificação de TEs

Um primeiro sistema de classificação de TEs foi proposto em 1989 por Finnegan, diferenciando os TEs de acordo com seu intermediário de transposição (Finnegan 1989). Os elementos de classe I ou retrotransposons realizam sua mobilização por um intermediário de RNA, ao passo que os elementos de classe II ou transposons de DNA não necessitam de um intermediário de RNA. Elementos de classe I realizam a transposição pelo mecanismo de transposição replicativa conhecido como 'copia-e-cola' (Meena e cols. 2012; Zhang e cols. 2014). Este mecanismo está relacionado com o aumento de genomas, inclusive com o gigantismo genômico observado em certas espécies de salamandras (Sotero-Caio e cols. 2017), por exemplo. O elemento se insere em uma posição diferente do genoma a partir de uma cópia transcrita, mantendo-se em sua posição original; após, a nova cópia sofre o processo de transcrição reversa por uma enzima transcriptase reversa produzida pelo próprio elemento ou por outro elemento da mesma classe, e o cDNA resultante insere-se na nova posição (Saleh e cols. 2019). Elementos de classe II em sua maioria se mobilizam pelo mecanismo de 'corta-e-cola' ou conservativo, no qual o elemento é removido do

sítio doador e se insere um sítio receptor pela enzima transposase, porém, alguns elementos desta classe podem se transpor pelo mecanismo de ‘copia-e-cola’ similar aos classe I, mas sem necessidade de transcrição reversa (Skipper e cols. 2013; Ochmann e Ivics 2021).

A grande diversidade de TEs trouxe à tona a necessidade de um sistema de classificação mais abrangente, como proposto por Wicker e cols. (2007), adicionando os níveis hierárquicos de subclasse, ordem, superfamília e família respectivamente (Tabela 1). A categoria de subclasse atualmente aplica-se aos elementos de classe II, separando-os em elementos que se transpõem pelo mecanismo de ‘corta-e-cola’ ou ‘descasca-e-cola’ (do inglês, *peel-and-paste*) (Di Stefano 2022). A divisão em Ordem se baseia no mecanismo de replicação e separa os elementos de classe I na ordem LTR, elementos com presença de longas repetições terminais (LTR) e ordens de elementos não-LTR, sendo estas DIRS, PLE, LINE e SINE; os elementos de classe II são separados nas ordens TIR, Crypton, Helitron e Maverick. Os elementos dentro de uma mesma ordem são separados em superfamília pelas diferenças estruturais no que concerne à presença de domínios ou organização enzimática, por exemplo (Piégu e cols. 2015; Anderson e cols. 2019). Os elementos das ordens Crypton, Helitron e Maverick apresentam apenas elementos homônimos à nível de superfamília. A divisão em famílias é feita baseada na similaridade das sequências e, adentrando níveis mais profundos de classificação, pode-se separar os elementos em subfamílias a partir de análise filogenética (Arkhipova 2017). Os TEs também são classificados de acordo com sua capacidade inerente de mobilização em elementos autônomos e não-autônomos, ambos os tipos podem ser de classe I ou classe II. Um TE autônomo é capaz de se transpor sozinho por codificar as enzimas necessárias para sua transposição. Os elementos não-autônomos dependem da maquinaria de elementos autônomos da mesma classe para se transporem (Wessler 2006; Lanciano e Cristofari 2020). Um exemplo ilustre é o complexo *Ac/Ds* identificado por Bárbara McClintock ao descobrir os TEs.

Tabela 1. Classificação hierárquica dos TEs de acordo com o modelo de Wicker e cols (2007).

Classe	Ordem	Superfamília
Classe I - Retrotransposons	LTR	*Copia*
		Gypsy
		Bel-Pao
		Retrovirus
		ERV
	DIRS	*DIRS*
		Ngaro
		VIPER
	LINE	*R2*
		RTE
		Jockey
		L1
	SINE	*tRNA*
		7SL
		5S
Classe II - transposons de DNA subclasse I	TIR	*Tc1-Mariner*
		hAT
		Mutator
		Merlin
		Transib
		P
		Piggyback
		CACTA
	Crypton	*Crypton*
Classe II - transposons de DNA subclasse II	Helitron	*Helitron*
	Maverick	*Maverick*

3. OBJETIVOS

3..1 Objetivo geral

Construir uma ferramenta amigável e acurada, baseada em deep learning, capaz de realizar a anotação e a classificação de elementos transponíveis em genomas eucarióticos.

3.2 Objetivos específicos

1. Criar conjuntos de dados curados de elementos transponíveis a partir de bancos de dados disponíveis;
2. Treinar modelos baseados em *deep learning* a partir dos conjuntos de dados para classificar elementos transponíveis;
3. Comparar a capacidade de classificação dos modelos com algumas das ferramentas mais usadas para a tarefa;
4. Criar um *workflow* para a anotação de elementos transponíveis usando os modelos de *deep learning* para a classificação;
5. Testar a ferramenta criada em genomas eucarióticos e comparar com as ferramentas de anotação mais comumente usadas.

4. Estrutura da tese

Esta tese está estruturada em forma de artigos. O primeiro artigo intitulado "*The good, the bad and the ugly about transposable elements annotation tools*" foi submetido a revista *Genetics and molecular biology* e está em fase de revisão. Neste manuscrito, fazemos um apanhado geral das principais ferramentas de bioinformática utilizadas para a classificação e anotação de TEs, e discutindo seus pontos fortes e fracos, assim como o que chamamos de "*ugly*", referindo-se a parte de usabilidade e documentação dos software, que é muitas vezes negligenciada pelos desenvolvedores. A partir desta ideia, buscamos apresentar soluções para este problema.

O segundo artigo apresenta o ponto principal desta tese: HamleTE, uma ferramenta desenvolvida para classificação e anotação de TEs usando modelos baseados em DL. Existem algumas ferramentas baseadas em DL para a classificação de TEs, porém nenhuma delas integra um workflow próprio para a anotação. Além disso, HamleTE propõe formas de refinar os resultados como seleção de valores de cut-off, tamanho de k-mers e otimizações para torná-lo capaz de ser rodado em máquinas de uso pessoal, sem a necessidade de servidores.

Há ainda um artigo em anexo resultante de um trabalho realizado durante o doutorado demonstrando a existência de mesmos supergrupos de *Wolbachia* em hospedeiros artrópodes de grupos taxonômicos diferentes, mesmo distantemente relacionados, o que não é esperado considerando-se o modo de transmissão vertical de *Wolbachia*. Isto evidencia a mudança de hospedeiros por transmissão horizontal, fenômeno que poucas vezes recebe a importância devida, mas possui papel fundamental na relação parasito-hospedeiro dado o impacto que *Wolbachia* tem em artrópodes.

CAPÍTULO 2

Artigo 1 - The good, the bad and the ugly of transposable elements annotation tools.

Tiago M. F. F. Gomes, Elverson S. de Melo, Gabriel L. Wallau, Elgion L. S. Loreto.

Submetido: 6 de maio de 2023
Revista: *Genetics and molecular biology*.
Manuscrito: GMB-2023-0138.

The good, the bad and the ugly of transposable elements annotation tools

Journal:	*Genetics and Molecular Biology*
Manuscript ID	Draft
Manuscript Type:	Review
Date Submitted by the Author:	n/a
Complete List of Authors:	Gomes, Tiago; Universidade Federal do Rio Grande do Sul, PPG Genética e Biologia Molecular Melo, Elverson; Aggeu Magalhaes Institute, Entomologia Wallau, Gabriel; Instituto Aggeu Magalhães – FIOCRUZ, Departamento de Entomologia; Aggeu Magalhaes Institute, Fundação Oswaldo Cruz Loreto, Elgion; Universidade Federal de Santa Maria, Bioquímica e Biologia Molecular;
Keyword:	transposable elements, bioinformatics, annotation, classification

The good, the bad and the ugly of transposable elements annotation tools

Tiago M. F. F. Gomes[1], Elverson S. de Melo[2], Gabriel L. Wallau[2], Elgion L. S. Loreto[1,3].

1. Programa de Pós-Graduação em Genética e Biologia Molecular, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, Brazil.

2. Departamento de Entomologia, Instituto Aggeu Magalhães, Fundação Oswaldo Cruz, Recife, Pernambuco, Brazil.

3. Biochemistry and Molecular Biology Department, Federal University of Santa Maria, Av. Roraima 1000, Santa Maria, RS, CEP 97105.900, Brazil.

TMFFG orcid; 0000-0002-1793-143X; ESM orcid: 0000-0003-3012-2503; GLW orcid:0000-0002-1419-5713; ELSL orcid: 0000-0002-7586-8168.

Corresponding author: Elgion Lucio da Silva Loreto. Universidade Federal de Santa Maria, Av. Roraima 1000. 97105-900, Santa Maria- RS- Brasil. elgionl@gmail.com.

ABSTRACT

Transposable elements are repetitive and mobile DNA segments that can be found in virtually all organisms investigated to date. Their complex structure and variable nature are particularly challenging from the genomic annotation point of view. Many softwares have been developed to automate and facilitate TEs annotation at a genomic scale, but they are highly heterogeneous regarding documentation, usability and methods. In this review, we revisited the existing softwares for TE genomic annotation, concentrating on the most often used softwares, the methodologies they apply, and user usability. Building on the state of the art of TE annotation softwares we propose best practices and highlight the strengths and weaknesses from the available solutions.

keywords: transposable elements, bioinformatics, annotation, classification.

INTRODUCTION

Transposable elements (TEs) are mobile genetic elements found in nearly every eukaryotic organism studied to date. As the name implies, these elements use the host molecular machinery to code their protein for mobilization. TEs are repetitive and sometimes fragmented, may be found within other TEs or protein-coding genes, and exhibit a wide range of structural, sequence-length, and distribution diversity. TEs constitute a significant portion of the genomes of many eukaryotic organisms, as for instance, 45% for humans and 85% in maize (Saleh et al. 2019; Stitzer et al. 2021; Hayward and Gilbert 2022). The method of transposition used by TEs varies depending on the TE class. Class I elements transpose via an RNA intermediate using a reverse transcriptase in what is known as "copy-and-paste" transposition; class II elements transpose via a DNA intermediate, with the majority of elements in this class using "cut-and-paste" mechanism, which is done by enzymes known as transposases (Wells and Feschotte 2020). TEs are yet subdivided in order, superfamily, family and subfamily (Wicker et al. 2007; Makałowski et al. 2019). In some species, *e.g. Homo sapiens*, despite having a high number of TEs, a few are known to be active, such as *ERV*s, *L1* and *Alu*, which are LTR and non-LTR class I elements, respectively (Ali et al. 2021; Autio et al. 2021). Furthermore, not all elements have the required machinery to transpose, and those lacking it are referred to as non-autonomous elements, relying on autonomous elements, which have the necessary enzymes to transpose. This is illustrated by the previously mentioned elements *L1* and *Alu* in humans, with the latter relying on the former to insert in a new location in a genome (Burns 2020; Chesnokova et al. 2022).

Using bioinformatics to find TEs in genomes is like putting together a puzzle with multiple copies of the same piece, each with its own place, some shredded or

with holes in it, and other pieces glued together with another piece. Choosing the right tools to solve the challenge of finding and classifying TEs in genomes is a difficult task, and there is currently no single tool that can thoroughly fulfill this effort on its own. Similarity-based, structure/motif pattern-matching, *de novo* prediction, or a workflow combining different methods are the approaches used by TEs annotation softwares, each with a trade-off between its strengths and weaknesses that need to be equated when choosing a program, that is, the good and the bad algorithmically speaking. There are two other frequently encountered software issues by researchers that we consider to be the "ugly" part: user friendliness and application development state.

Many of the most commonly used applications are not well maintained, failing to keep up with operating system updates or advances in the programming languages in which they are written, resulting in difficult installation due to obsolete dependencies required by the software. The problem of finding and installing the correct package versions can be overcome by using programs to create virtual environments or "containers". However, this does not guarantee that the required dependency versions will be available or that it will be easier to install. Another option is to compile either the software or its dependencies from source, which may result in a time-consuming snowball effect of finding software dependencies, all of which must be compatible with the operating system used.

To complete the task of installing and using the softwares, the human side must be considered. It necessitates skills that, depending on the researcher's background, may outweigh his or her knowledge or willingness to use the software. In line with this, not all softwares has a complete and clear documentation on how to run them and what the available options mean.

Herein, we bring to light the good, the bad and the ugly sides of using bioinformatics tools for genomic annotation of transposable elements. What are the most commonly used softwares, how to distinguish between methods and what can be done to advance the current state-of-the-art on the subject.

METHODS AND SOFTWARES FOR TE ANNOTATION

The process of detecting a TE sequence in a genome, classifying it, and identifying its coordinates, *i.e.* the start and end of a sequence, in a chromosome or in contigs is referred to as TE annotation. The repetitiveness of TEs, the number of very similar or degraded copies, and the presence of nested elements are some of the challenges faced by TE annotation softwares. Tools designed to annotate TEs may use sequence similarity, the presence of structural elements such as long terminal repeats (LTR) or terminal inverted repeats (TIR), and a *de novo* approach to accomplish this task.

Table 1 summarizes the main features of the softwares used for TE annotation and classification, such as the release year, method for TE characterization, the software development status, *i.e.*, whether it is still receiving updates, improvements, or developer support, and other aspects such as the operating system required to run the software if it is a downloadable version.

Similarity-based

The most used method for characterizing sequences in general (Zielezinski et al. 2017; Carey et al. 2021), many times wrongly named homology-based (Reeck et al. 1987; Pearson 2013). It is used by RepeatMasker (Smit et al. 2013) and CENSOR (Kohany et al. 2006), two of the most well-known and widely used tools for masking

repetitive sequences (Figure 1a). Similarity-based searches have high specificity and accuracy, making it useful for detecting conserved regions of related sequences, single nucleotide polymorphisms, and indels. Disadvantages are its computational complexity, it may not work well with highly divergent sequences, can generate false positives when working with repetitive sequences as TEs, and are limited to known sequences, i.e. does not allow for the discovery of new TEs.

RepeatMasker searches genomic data for interspersed repeats and low complexity DNA sequences, by default using the Dfam database as queries including Hidden Markov Models profiles and consensus sequences (Storer et al. 2021), but it is also possible to use a Repbase-like formatted custom library instead. RepeatMasker is written in Perl, an interpreted programming language, meaning it does not need to be compiled from source, it includes installation instructions, basic usage and a detailed program manual with all of the information needed regarding all parameters. It can be installed from the bioconda channel in a conda virtual environment. RepeatMasker is still maintained, updated, and has news about newer releases on its website. It is an open-source software available for download at https://www.repeatmasker.org/ or https://github.com/rmhubley/RepeatMasker.

CENSOR compares nucleotide or amino acid sequences to known repeats using WU-BLAST (in newer paid versions there is an option to use BLAST instead), and can compare sequences of DNA-DNA or DNA-protein. CENSOR is available as a web-based service or standalone program to be used in UNIX systems. The web version uses the REPBASE database, which for download needs a paid subscription since 2018. The standalone version available for download (at https://www.girinst.org/downloads/software/censor) was last updated in 2016, has a short description on how to use and no manual describing the options.

Structure-based

Tools that search for structure in sequences can discover catalytic sites and functional protein sites. It can also be used to improve similarity-based alignment results. This method is limited by the availability of known sequence structures and does not work well with highly variable regions or homologs that are highly divergent. Two of the most used softwares using this method are LTR_finder (Xu and Wang 2007) and MITE-hunter (Han and Wessler 2010), as other tools as TIRmite (found at https://github.com/Adamtaranto/TIRmite).

LTR_finder identifies full-length LTR elements in genomic data by searching possible exactly matching pairs at the 5' and 3' end of sequences, selecting the pairs based on a specified distance between them, calculates the similarity between regions using global alignment and adjusts the near-end boundaries using the Smith-Waterman algorithm. It is presented both as a web-server and a standalone version for UNIX systems. The latter is written in C and C++ and must be compiled from the source code. It is also dependent on Perl. The manual makes no mention of dependency versions or the requirement to install the Perl module GD, which is required for bitmap handling. The LTR_finder repository on github (https://github.com/xzhub/LTR_Finder) is not maintained anymore and the webserver (http://tlife.fudan.edu.cn/ltr_finder/), to the moment of this writing, was not available.

MITE-hunter is a program that searches for miniature inverted-repeat transposable elements (MITEs), which are short non-autonomous Class II elements found in plants and animals. MITE-hunter is written in Perl and is intended to run on UNIX systems. It first identifies candidates based on the presence or absence of TIRs and target site duplications (TSDs), then performs an all-by-all BLASTN comparison

to filter false positives and clusters selected sequences. A multiple sequence alignment is performed to generate consensus sequences, which are then categorized into families. It can be downloaded on http://target.iplantcollaborative.org/mite_hunter.html, but does not appear to be in development any longer, as the last update on its github page (https://github.com/jburnette/MITE-Hunter) was in 2010. MITE-hunter depends on NCBI BLAST, Muscle, mDust and the Perl programming language to be installed and used. The manual makes no mention of the dependencies versions.

De novo

The *de novo* method does not require a reference database to find TEs, which is useful when working with newly sequenced genomes. Conversely, it can produce unreliable results due to sequencing or assembling errors, and because there are no curated sequences as reference to validate the results. It usually works by performing an all-by-all sequence comparison followed by sequence clustering or by directly applying clustering methods to reads that will be downsampled or filtered (Storer et al. 2022). RepeatModeler (Smit and Hubley 2008), EDTA (Ou et al. 2019) and LTR annotator (You et al. 2015) are some examples of tools using this method, being RepeatModeler and EDTA two of the most used.

RepeatModeler is a pipeline for de novo TE identification that aims to produce a reliable and consistent TE library of consensus sequences of unique TE families. It uses Recon for repeat discovery, which uses a sensitive alignment approach and is well suited to discovering old TE families, and RepeatScout, which is faster and detects the most abundant and younger families more easily. RepeatModeler is mostly written in Perl, having a complete and detailed manual on how to install and run it, with

all of its dependencies clearly specified with the necessary versions. It is still maintained and is available at https://github.com/Dfam-consortium/RepeatModeler or http://www.repeatmasker.org/RepeatModeler/. The newer version RepeatModeler2 (Flynn et al. 2020) integrates a structure discovery step of LTR elements to improve the discovery of elements of this class.

EDTA is a package designed for *de novo* TE annotation that aims to generate a high-quality non-redundant TE library for whole sequenced genomes. It was developed by benchmarking many TE tools using a manually curated rice TE library, and selecting the most performant ones to be part of the TE annotation pipeline, which includes LTRharvest, a parallel version of LTR_FINDER, LTR_retriever, GRF, TIR-Learner, HelitronScanner, and RepeatModeler. EDTA is written using Perl, Python and shell script, and can be installed using a conda virtual environment, singularity or docker containers. Its manual contains detailed descriptions on how to install and run the program, as well as information on the input and output files. It is still maintained and updated, being found at https://github.com/oushujun/EDTA. It can also be used to test new TE annotation methods or TE libraries using the rice genome, according to the authors of EDTA. The input FASTA sequence identifiers (IDs) must be at most 13 characters long, and many non-alphanumeric characters are not permitted; otherwise, the program execution is terminated. There is no tool or script included with the package to edit the invalid IDs, leaving it up to the user to do so.

Combined approaches

Because TEs are such complex elements with so many features to consider in order to correctly annotate them, the scientific community has agreed that a

combination of *de novo*, similarity, and structure-based approaches is the best strategy for a more careful and accurate characterization of TEs. TIR-learner (Su et al. 2019), REPET (Flutre et al. 2011), DAWGPAWS (Estill and Bennetzen 2009) and Earl Grey (Baril et al. 2022) are examples of such tools.

TIR-learner is a tool developed to detect TIRs primarily in plant genomes and is available at https://github.com/WeijiaSu/TIR-element-annotation. It uses a pipeline of combining similarity and structure approaches with a *de novo* structure screening, which uses a machine learning algorithm to classify sequences into five TIR superfamilies. Next, it removes overlaps by comparing the outputs of each method, resulting in a library of TIR-elements. It is written in Python and shell script, and it is dependent on the softwares Generic Repeat Finder (GRF) and BLAST+. It includes a simple and straightforward manual for installing and running the software. There is no mention of specific version dependencies. Its most recent version is 1.14, which was updated in 2019 with newer unresolved github issues.

REPET is a software suite that uses two main pipelines to annotate TEs at the genomic scale: TEdenovo and TEannot. The former compares a genome to itself using BLASTER and then clusters the resulting matches using GROUPER, RECON, and PILER. For each cluster, a multiple sequence alignment is performed in order to construct a consensus sequence and then classify it. After that, TEannot combines multiple programs to reconstruct intact TE copies and filter out fragmented copies and false-positives. REPET is written in C++ and Python to be used in Linux-based systems, it depends on several external programs, with some dependency versions being deprecated or not yet maintained upstream, such as the required Python version (version 2.x). To help address those issues, there is a docker version. The REPET manual has detailed information about software versions, installation and usage. It is

still maintained, with recent updates on its containerized version, PFAM database and a newly added eukaryotic rRNA database. The REPET package and its instructions can be found at http://urgi.versailles.inra.fr/Tools/REPET.

Classifiers

Following the step of generating a series of TE consensus, the newly created library must be classified, which will give those sequences meaning. Although many TE annotation pipelines rely on some sort of classification mechanism (Flutre et al. 2011; Flynn et al. 2020; Riehl et al. 2022), this mechanism does not always follow a classification scheme adopted by a research group, or provide the level of detail desired by the researcher. Furthermore, different classifiers generate predictions using different databases as a source of comparison. The distribution of TE types in a database, as well as the divergence between the species under study and the species present in the database, will have a direct impact on the classification quality, because there is a loss of TE identification when very divergent reference sequences are used (Bell et al. 2022). It is also known that different classification methods have varying accuracies, with some better classifying specific groups of TEs than others (Hoede et al. 2014; Monat et al. 2016; Zhang et al. 2022). As a result, it is frequently necessary to apply multiple classification methods to a newly created library in order to resolve ambiguities in more divergent consensus (Melo and Wallau 2020).

In recent years, TE classification mechanisms have evolved significantly. In general, they can be divided into two large groups (Figure 1b): I) programs that employ traditional approaches, such as the use of various types of blasts and search algorithms for protein domains like HMMER. including REPCLASS (Feschotte et al. 2009), PASTEC (Hoede et al. 2014), RepeatClassifier (a classification program from

RepeatModeler 2), LTRclassifier (Monat et al. 2016), TEsorter (Zhang et al. 2022) and RTclass1 (Kapitonov et al. 2009); II) programs that use machine learning algorithms, including TEclass (Abrusán et al. 2009), DeepTE (Yan et al. 2020), ClassifyTE (Panta et al. 2021) and TERL (da Cruz et al. 2021).

One of the most cited classifiers is PASTEC. It is part of the REPET pipeline and thus has the same set of manuals, whether it is installed alongside the main package or used within a container provided by the developers. PASTEC searches sequences for structural features such as TIRs or LTRs, as well as the presence of SSRs, ORFs, and poly(A) tails. This program also searches for sequence similarity against Repbase sequences and Pfam domains. One of the most intriguing aspects of PASTEC is its user-friendly output, which includes a tabular file with a classification combined with a confidence index for each sequence, as well as lists of structural characteristics, protein domains, and blast matches against Repbase. Despite this, as there is no longer free access to Repbase, the library used by PASTEC has become outdated. REPCLASS employs a similar strategy, but their software has not been updated in at least 8 years, and has WU-blast, a discontinued program, as a dependency. RepeatClassifier (installed with RepeatModeler) can use Dfam as the database for its classification task, circumventing the challenge of accessing up-to-date data from Repbase. However, the output of this software is very streamlined, consisting only of a multi-fasta file containing the TE classification in the original sequence header.

While all three of these tools are designed to categorize TEs of any kind, some tools concentrate on doing so in greater detail. Both TEsorter and RTclass1 can classify LTRs and LINEs at the clade level. RTclass1, a Repbase database service, can classify TE at the clade level in seconds; the user only needs to supply the amino

acid sequence of the TE protein's reverse transcriptase domain. Despite being easy, it only works for non-LTR TEs. TEsorter, like most TE-related programs, requires a local installation; however, it is quite simple to install using the conda package manager. This software compares translated TE sequences to profiles in GypsyDB and RexDB. However, while it can generate a classification for any type of TE, it can only classify LTR-type TEs at the clade level. TEclass was one of the first classifiers to use machine learning algorithms. It was last updated in 2016, when the Random Forest and LVQ algorithms were added to the SVM algorithm that had previously been used in the classifier's first version. In addition to the local installation option, it also provides the option to run the analyses on a web server, making it easier to use for less experienced users. Despite this, the program can only classify TEs into one of four major groups: DNA, LINE, LTR or SINE.

This limitation was recently overcome by DeepTE, ClassifyTE, and TERL, which also use machine learning (usually artificial neural networks) to classify TEs at the superfamily level. These three programs all run only locally, requiring installation, which may be difficult for some users. Another issue the three programs have in common is that they all generate only one classification label for each sequence, even though their output structures differ. TERL, for example, replaces a sequence's entire header with its classification label, making it difficult for the user to manage multi-fasta files. There is also no information about the accuracy of each class prediction in any of these three programs. Furthermore, other factors can have an impact on the user experience. For example, ClassifyTE requires that the TE library that needs to be classified be located in the "data" folder in the application's root directory, which can limit the application's flexibility.

DISCUSSION

In the quest to better understand and unravel the complexity of life from a genomic perspective, bioinformatics has become an indispensable ally of geneticists and molecular biologists. The exponential availability of genomic data creates an increasing need for the development of tools capable of balancing efficiency and ease of use, preventing either from becoming a hindrance to research. Because of a plethora of genetic and structural features that make correct annotation difficult, TEs add another dimension to this picture. To undertake such hardships, many strategies are employed to detect and characterize TEs on genomes.

Similarity-based tools (RepeatMasker, CENSOR) employ a well-established method that uses libraries or sets of known sequences that for an increasing number of species have experimental validation, generating precise results. The bad side is that it depends on the reliability of the dataset used as a library, its efficiency and precision can quickly decrease when used to detect, for example, protein sequences with only a few distinct residues and is time demanding and memory consuming (Zielezinski et al. 2017).

Structure-based methods, such as LTR_FINDER and MITE-hunter, are best-tailored to detect protein domains or class-specific patterns of TE sequences. The search strategy behind structure-based methods is either an enumerative approach, where sequences are analyzed as small words contained in the query and then compared to a collection of patterns, or probabilistic, in which patterns are searched using a motif or a weighted matrix (Hashim et al. 2019). Equally to similarity-based tools, the search time increases as the dataset grows and is also dependent on known patterns. Nonetheless, when compared to the amount of TE libraries for use with similarity-based tools, there are even less structures/motifs available.

RepeatModeler and EDTA, for example, use the *de novo* methodology to annotate TEs, which is effective for discovering novel TE sequences and creating a non-redundant TE collection. Most of the time, *de novo* tools operate by automatically comparing sequences and grouping those that share the most similarities (Storer et al. 2022). The disadvantage of this method is that it produces more false-positive results than other approaches, is more likely to result in chimeric sequences, and may make it more difficult to distinguish between different TE fragments, sometimes even including pieces of non-TE sequences like those from repetitive gene families from the host genome.

Combining strategies is currently a scientific consensus as a way to minimize the drawbacks of a technique while maximizing its benefits (Arkhipova 2017). Nonetheless, combining methods brings its own problems to the game. Combining methods also entails combining the disparate output of each program, analyzing the results, removing redundant but not necessarily identical TE sequences, and typically clustering the results. All of this takes more time and computational resources to run, and it does not solve the problem of redundant sequences being classified with different labels. That is why understanding how each method works, as well as the benefits and drawbacks of the tools used, is critical to knowing what results to expect from the annotation.

Regardless of the good and bad of each software's methodology, if it is unclear what is needed to install it, how to use it, and how comprehensible the output is, the researcher may opt to avoid using cutting-edge or more performant software in favor of older but better documented tools. In other words, when annotating TEs, or even in bioinformatics in general, user friendliness and documentation completeness must be considered.

A poorly documented software may lead the daily work of a researcher to setbacks and delays, by adding a new layer of complexity to the already complex task of working with biological data (Lawlor and Sleator 2020). It would be similar to conducting a wet lab experiment without fully understanding the chemicals, their activities, or not having the label's information regarding concentration. It is especially true for small research groups or underfunded institutions that do not have enough financial support to hire a specialist to work on the task, which can become, at a certain level, an obstacle to progress in their field of study and to keep pace with the state-of-the-art (Krampis 2022). If the quality of software documentation was evaluated as carefully as other topics in peer-reviewed papers on bioinformatics tools, it could contribute to better documented softwares. Karimzadeh and Hoffman (2018) propose guidelines for creating good software documentation, including, as minimum requirements, a page with code and an issue tracker (*e.g.* Github and Gitlab), a "Readme" file containing the main points for installation and usage, and a manual with a detailed description of every parameter.

It is not uncommon for TE annotation softwares to use discontinued or outdated packages, causing installation and usage issues, as well as becoming a bottleneck to computer performance, which goes against the ever-increasing computer power and technological advances in operating systems and programming languages. It may also occur as a result of the software's development being halted and becoming an *abandonware*, not receiving any upgrades, also affecting the developer's error support for users. Another issue is retro-fitting older tools to new conditions, *i.e.,* a tool developed to identify a certain feature may be unable to extract all the correct information obtained by newer research leading to incomplete results. (Lawlor and Walsh 2015). The software installation can be impacted by outdated packages,

whether it is because the required program does not have a version for more recent operating systems or because the software's dependencies cannot be installed. Attempting to install outdated versions on newer platforms may result in version conflicts, leading to the "dependency hell," a frustrating situation in which a software cannot be utilized due to incompatibilities between softwares with shared packages but that need different versions, particularly for softwares that require a large number of packages.

Virtual environments and containers are methods for dealing with dependency issues, allowing programs to run on any system (Krampis 2022). Version conflicts can, however, still occur in virtual environments such as Conda. Containers are more reliable in this regard because they provide a more isolated environment due to operating system-level virtualization, but may be trickier to set up. Dockerfiles and Conda recipes, files containing all commands and software versions to automatically assemble a container or create a virtual environment, make software installation easier, aid in experiment reproducibility and avoid dealing with dependency issues that may arise when manually installing software and looking for its dependencies.

Lack of documentation, software updates and developer support are examples of the “ugly side” of TEs annotation tools and bioinformatics as a whole, all of which are unrelated to the method’s good and bad. On top of that, the ugly side may enter the picture when a developer creates a program to solve their own research problem, releases it for the scientific community, but does not fully adapt it for general use, casting aside good software development practices for what worked on the original project. Parameters and outputs that appear clear to the developer may be confusing to the end-user, making the program less user friendly and less understandable for biologists or other life scientists, who are the best suited to validate the findings (Lerat

2010). In an ideal world, a biologist would have the skills of a software engineer and vice versa, however this is far from the reality due to the complexity of both disciplines. The adoption of best practices for developing and deploying bioinformatics software, along with software documentation that adheres to guidelines to better inform the user, would provide a solid foundation for improving TEs annotation tools and the standard of related research (Lawlor and Walsh 2015; Lawlor and Sleator 2020). The creation of better documented and user-friendly tools can be aided by initiatives like TE Hub (Elliott et al. 2021), a collaborative platform that aims to provide information for the TE scientific community with a focus on databases, tools, and methods for TE annotation. TE Hub offers a way to integrate information and standardize protocols for tools related to TE scientific research. Figure 2 depicts a score for the tools mentioned here based on the availability or absence of several types of documentation, such as a reference manual, an informative figure illustrating how the software works, and whether there is an alternative method of installation other than manual installation. Table S1 contains a more detailed version that shows what features are present or absent for each software.

Therefore, when choosing a TE annotation software the researchers should always ask themselves: is this the best tool for my needs? What are the downsides? Is the documentation clear about what is required to use the software? Is the software still being actively developed/maintained? Does the developer provide user support? These questions might seem simple, but given the significance of knowing how to get the most out of a tool, they help to achieve better research results, particularly in terms of software usability. Having a reliable TE annotation is the ultimate goal. This can be accomplished by improving the status of existing tools, which calls for both end-user and developer effort. For that, the user requires better documented tools as well as a

place to share information with the developer so that the developer knows what to do to create a more well-known tool, benefiting the entire TEs scientific community.

Conflict of interest

The authors declare that there is no conflict of interest that could be perceived as prejudicial to the impartiality of the reported research.

Authors Contributions

TMFFG, GLW and ELSL conceived the study; TMFFG and ESM conducted the experiments and analyzed the data; TMFFG, ESM, GLW and ELSL wrote the manuscript. All authors read and approved the final version.

References

Abrusán G, Grundmann N, DeMester L and Makalowski W (2009) TEclass—a tool for automated classification of unknown eukaryotic transposable elements. Bioinformatics 25:1329–1330.

Ali A, Han K and Liang P (2021) Role of Transposable Elements in Gene Regulation in the Human Genome. Life 11:118.

Arkhipova IR (2017) Using bioinformatic and phylogenetic approaches to classify transposable elements and understand their complex evolutionary histories. Mob DNA 8:19.

Autio MI, Bin Amin T, Perrin A, Wong JY, Foo RS-Y and Prabhakar S (2021) Transposable elements that have recently been mobile in the human genome. BMC Genomics 22:789.

Baril T, Imrie RM and Hayward A (2022) Earl Grey: a fully automated user-friendly transposable element annotation and analysis pipeline. doi: 10.21203/rs.3.rs-1812599/v1

Bell EA, Butler CL, Oliveira C, Marburger S, Yant L and Taylor MI (2022) Transposable element annotation in non-model species: The benefits of species-specific repeat libraries using semi-automated EDTA and DeepTE de novo pipelines. Mol Ecol Resour 22:823–833.

Burns KH (2020) Our Conflict with Transposable Elements and Its Implications for Human Disease. Annu Rev Pathol Mech Dis 15:51–70.

Carey KM, Patterson G and Wheeler TJ (2021) Transposable element subfamily annotation has a reproducibility problem. Mob DNA 12:4.

Chesnokova E, Beletskiy A and Kolosov P (2022) The Role of Transposable Elements of the Human Genome in Neuronal Function and Pathology. Int J Mol Sci 23:5847.

da Cruz MHP, Domingues DS, Saito PTM, Paschoal AR and Bugatti PH (2021) TERL: classification of transposable elements by convolutional neural networks. Brief Bioinform 22:bbaa185.

Elliott TA, Heitkam T, Hubley R, Quesneville H, Suh A, Wheeler TJ, and The TE Hub Consortium (2021) TE Hub: A community-oriented space for sharing and connecting tools, data, resources, and methods for transposable element annotation. Mob DNA 12:16.

Estill JC and Bennetzen JL (2009) The DAWGPAWS pipeline for the annotation of genes and transposable elements in plant genomes. Plant Methods 5:8.

Feschotte C, Keswani U, Ranganathan N, Guibotsy ML and Levine D (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.

Flutre T, Duprat E, Feuillet C and Quesneville H (2011) Considering Transposable Element Diversification in De Novo Annotation Approaches. PLOS ONE 6:e16526.

Flynn JM, Hubley R, Goubert C, Rosen J, Clark AG, Feschotte C and Smit AF (2020)

RepeatModeler2 for automated genomic discovery of transposable element families. Proc Natl Acad Sci 117:9451–9457.

Han Y and Wessler SR (2010) MITE-Hunter: a program for discovering miniature inverted-repeat transposable elements from genomic sequences. Nucleic Acids Res 38:e199.

Hashim FA, Mabrouk MS and Al-Atabany W (2019) Review of Different Sequence Motif Finding Algorithms. Avicenna J Med Biotechnol 11:130–148.

Hayward A and Gilbert C (2022) Transposable elements. Curr Biol 32:R904–R909.

Hoede C, Arnoux S, Moisset M, Chaumier T, Inizan O, Jamilloux V and Quesneville H (2014) PASTEC: An Automatic Transposable Element Classification Tool. PLoS ONE 9:e91929.

Kapitonov VV, Tempel S and Jurka J (2009) Simple and fast classification of non-LTR retrotransposons based on phylogeny of their RT domain protein sequences. Gene 448:207–213.

Karimzadeh M and Hoffman MM (2018) Top considerations for creating bioinformatics software documentation. Brief Bioinform 19:693–699.

Kohany O, Gentles AJ, Hankus L and Jurka J (2006) Annotation, submission and screening of repetitive elements in Repbase: RepbaseSubmitter and Censor. BMC Bioinformatics 7:474.

Krampis K (2022) Democratizing bioinformatics through easily accessible software platforms for non-experts in the field. BioTechniques 72:36–38.

Lawlor B and Sleator RD (2020) The democratization of bioinformatics: A software engineering perspective. GigaScience 9:giaa063.

Lawlor B and Walsh P (2015) Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. Bioengineered 6:193–203.

Lerat E (2010) Identifying repeats and transposable elements in sequenced genomes: how to find your way through the dense forest of programs. Heredity 104:520–533.

Makałowski W, Gotea V, Pande A and Makałowska I (2019) Transposable Elements: Classification, Identification, and Their Use As a Tool For Comparative Genomics. In: Anisimova M (ed) Evolutionary Genomics: Statistical and Computational Methods. Springer, New York, NY, pp 177–207

Melo ES de and Wallau GL (2020) Mosquito genomes are frequently invaded by transposable elements through horizontal transfer. PLOS Genet 16:e1008946.

Monat C, Tando N, Tranchant-Dubreuil C and Sabot F (2016) LTRclassifier: A website for fast structural LTR retrotransposons classification in plants. Mob Genet Elem 6:e1241050.

Ou S, Su W, Liao Y, Chougule K, Agda JRA, Hellinga AJ, Lugo CSB, Elliott TA, Ware D,

Peterson T et al. (2019) Benchmarking transposable element annotation methods for creation of a streamlined, comprehensive pipeline. Genome Biol 20:275.

Panta M, Mishra A, Hoque MT and Atallah J (2021) ClassifyTE: a stacking-based prediction of hierarchical classification of transposable elements. Bioinforma Oxf Engl 37:2529–2536.

Pearson WR (2013) An Introduction to Sequence Similarity ("Homology") Searching. Curr Protoc Bioinforma Ed Board Andreas Baxevanis Al 0 3:10.1002/0471250953.bi0301s42.

Reeck GR, de Haën C, Teller DC, Doolittle RF, Fitch WM, Dickerson RE, Chambon P, McLachlan AD, Margoliash E, Jukes TH et al. (1987) "Homology" in proteins and nucleic acids: A terminology muddle and a way out of it. Cell 50:667.

Riehl K, Riccio C, Miska EA and Hemberg M (2022) TransposonUltimate: software for transposon classification, annotation and detection. Nucleic Acids Res 50:e64.

Saleh A, Macia A and Muotri AR (2019) Transposable Elements, Inflammation, and Neurological Disease. Front. Neurol. 10:

Smit A and Hubley R (2008) RepeatModeler Open-1.0.

Smit A, Hubley R and Green P (2013) RepeatMasker Open-4.0.

Stitzer MC, Anderson SN, Springer NM and Ross-Ibarra J (2021) The genomic ecosystem of transposable elements in maize. PLoS Genet 17:e1009768.

Storer J, Hubley R, Rosen J, Wheeler TJ and Smit AF (2021) The Dfam community resource of transposable element families, sequence models, and genome annotations. Mob DNA 12:2.

Storer JM, Hubley R, Rosen J and Smit AFA (2022) Methodologies for the De novo Discovery of Transposable Element Families. Genes 13:709.

Su W, Gu X and Peterson T (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.

Wells JN and Feschotte C (2020) A Field Guide to Eukaryotic Transposable Elements. Annu Rev Genet 54:539–561.

Wicker T, Sabot F, Hua-Van A, Bennetzen JL, Capy P, Chalhoub B, Flavell A, Leroy P, Morgante M, Panaud O 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:W265-268.

Yan H, Bombarely A and Li S (2020) DeepTE: a computational method for de novo classification of transposons with convolutional neural network. Bioinformatics 36:4269–4275.

You FM, Cloutier S, Shan Y and Ragupathy R (2015) LTR Annotator: Automated Identification and Annotation of LTR Retrotransposons in Plant Genomes. Int J Biosci Biochem Bioinforma 5:165–174.

Zhang R-G, Li G-Y, Wang X-L, Dainat J, Wang Z-X, Ou S and Ma Y (2022) TEsorter: An accurate and fast method to classify LTR-retrotransposons in plant genomes. Hortic Res 9:uhac017.

Zielezinski A, Vinga S, Almeida J and Karlowski WM (2017) Alignment-free sequence comparison: benefits, applications, and tools. Genome Biol 18:186.

Figure legends

Figure 1. Schematic representation of some softwares available for TE annotation (a) and classification (b) based on the method for TE detection.

Figure 2. Software score for annotators (a) and classifiers (b) based on documentation availability. The final score, which ranges from 0 to 1, is determined by the presence or absence of various types of documentation, such as a manuscript, reference manual, Readme file, quick start section, informative figure demonstrating how the software works, frequently asked questions (FAQ), news section, issue tracker, and built-in help.

--

Supplementary material - the following online material is available for this article:

Table S1 - Presence or absence of types of documentation by software.

Table 1. Summary of the key features of tools used to annotate or classify TEs.

Software	Year	Type	Method	Development status	Presentation	Operating system	Installation	External dependencies	Alternative installation
CENSOR	1996	Annotator	Similarity	Maintained*	Web/Downloadable	-	-	No	-
ClassifyTE	2021	Classifier	Machine learning	Maintened	Downloadable	Unix	Executable script	Yes	Venv
DAWGPAWS	2009	Annotator	Combined	Not maintained	Downloadable	Unix	Executable script	Yes	-
DeepTE	2020	Classifier	Machine learning	Maintained	Downloadable	Unix	Executable script	Yes	Venv
EarlGrey	2022	Annotator	Combined	Maintained	Downloadable	Linux	Executable script	Yes	Venv/Container
EDTA	2019	Annotator	*De novo*	Maintened	Downloadable	Linux	Executable script	Yes	Venv/Container
LTR annotator	2015	Annotator	*De novo*	Not maintained	Downloadable	Linux	Executable script	Yes	-
LTR classifier	2016	Classifier	Library-based	Maintained	Web	-	-	-	-
LTR_finder	2007	Annotator	Structure	Not maintained	Downloadable	Linux	Source code	No	-
MITE-hunter	2010	Annotator	Structure	Not maintained	Downloadable	Linux	Executable script	No	-
PASTEC	2014	Classifier	Library-based	Maintained	Downloadable	Linux	Executable script	Yes	Container
reasonaTE	2022	Annotator	Combined	Maintained	Downloadable	Linux	Executable script	Yes	Venv
REPCLASS	2015	Classifier	Library-based	Not maintained	Downloadable	Linux	Executable script	Yes	-
RepeatClassifier	2020	Classifier	Library-based	Maintaned	Downloadable	Linux	Executable script	-	Venv
RepeatMasker	1997	Annotator	Similarity	Maintained	Web/Downloadable	Linux	Executable script	Yes	Venv
RepeatModeler	2008/2020	Annotator	*De novo*	Maintained	Downloadable	Linux	Executable script	Yes	Venv
REPET	2011	Annotator	Combined	Maintained	Downloadable	Linux	Executable script	Yes	Container
RTclass1	2010	Classifier	Library-based	Maintained	Web/Downloadable	Linux	Executable script	Yes	-
TERL	2020	Classifier	Machine learning	Maintained	Downloadable	Unix	Executable script	No	Venv
TEsorter	2022	Classifier	Library-based	Maintained	Downloadable	Unix	Executable script	Yes	Venv
TIR-learner	2019	Annotator	Combined	Maintained	Downloadable	Linux	Executable script	Yes	-
TIRmite	2017	Annotator	Structure	Maintained	Downloadable	Linux	Executable script	Yes	Venv

Legend. Executable script: the program does not need to be compiled from source code; Source code: program needs to be compiled to install; Venv: program can be installed using a virtual environment; Container: program can be installed as a container using tools such as Docker or Singularity. Linux: program needs to be installed in a Linux based operating system (OS); Unix: a Unix-based operating system such as MacOS or a Linux-based OS; External dependencies: tools mandatory to run the main program that are not installed in the main program installation. *: just the web version appears to be maintained.

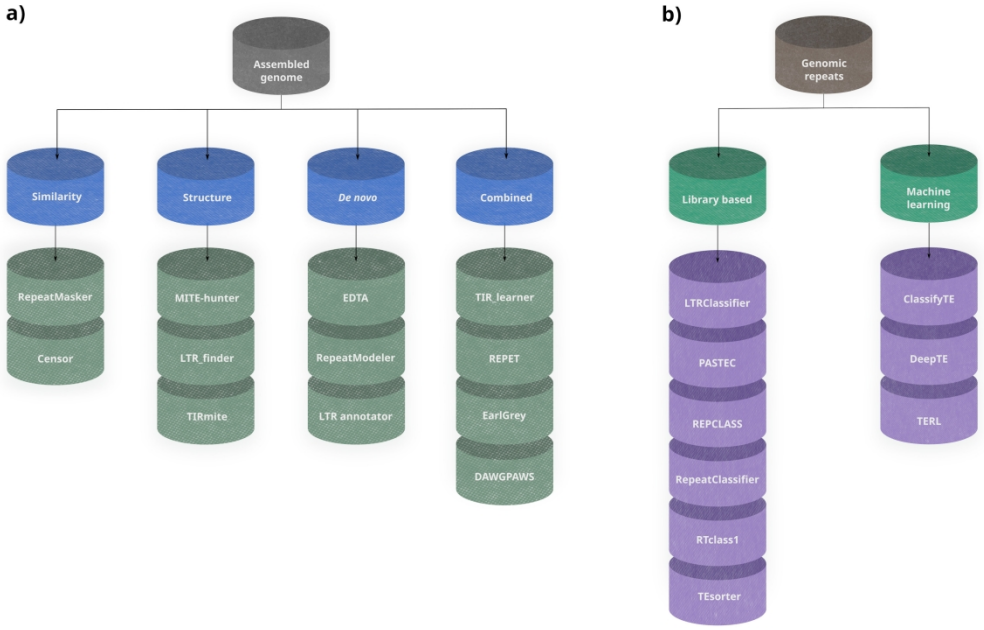


Figure 1. Schematic representation of some softwares available for TE annotation (a) and classification (b) based on the method for TE detection.

291x187mm (300 x 300 DPI)

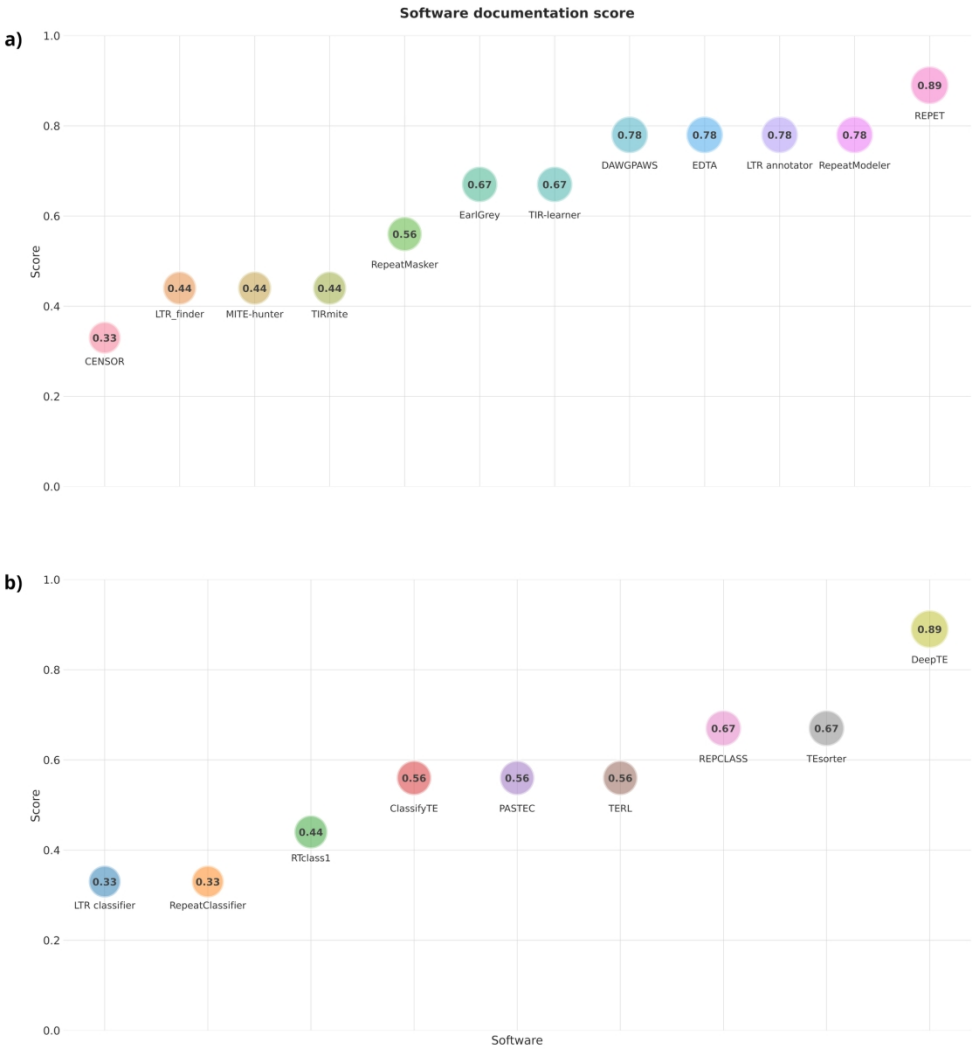


Figure 2. Software score for annotators (a) and classifiers (b) based on documentation availability. The final score, which ranges from 0 to 1, is determined by the presence or absence of various types of documentation, such as a manuscript, reference manual, Readme file, quick start section, informative figure demonstrating how the software works, frequently asked questions (FAQ), news section, issue tracker, and built-in help

177x189mm (300 x 300 DPI)

Table S1. Presence or absence of types of documentation by software.

Software	Manuscript	Reference Manual	Readme	Quick start	Informative figures	FAQ	News	Issue tracker	Built-in help	Score
CENSOR	1	0	0	0	0	0	1	0	1	0.33
ClassifyTE	1	0	1	1	0	0	0	1	1	0.56
DAWGPAWS	1	1	1	1	0	0	1	1	1	0.78
DeepTE	1	0	1	1	1	1	1	1	1	0.89
EarlGrey	1	0	1	1	1	0	0	1	1	0.67
EDTA	1	1	1	1	1	0	0	1	1	0.78
LTR annotator	1	1	1	1	1	0	1	0	1	0.78
LTR classifier	1	0	0	0	1	0	0	0	1	0.33
LTR_finder	1	0	1	0	0	0	0	1	1	0.44
MITE-hunter	1	1	0	0	0	0	0	1	1	0.44
PASTEC	1	1	1	1	0	0	0	0	1	0.56
reasonaTE	1	0	1	1	1	0	0	1	1	0.67
REPCLASS	1	0	1	1	0	0	1	1	1	0.67
RepeatClassifier	1	0	0	0	0	0	0	1	1	0.33
RepeatMasker	0	0	1	0	0	1	1	1	1	0.56
RepeatModeler	1	1	1	1	0	0	1	1	1	0.78
REPET	1	1	1	0	1	1	1	1	1	0.89
RTclass1	1	0	0	0	1	0	1	0	1	0.44
TERL	1	0	1	1	0	0	0	1	1	0.56
TEsorter	1	0	1	1	1	0	0	1	1	0.67
TIR-learner	1	0	1	1	1	0	0	1	1	0.67
TIRmite	0	0	1	1	0	0	0	1	1	0.44

Legend

0	The absence of a feature.
1	The presence of a feature.
Manuscript	Conceptual and technical details of the method.
Reference manual	Complete details of every configurable setting, input and output.
Readme	Basic instructions for installation and use of the software and where to find more information. Describe how to install your software and all of its dependencies, in detail.
Quick start	Step-by-step instructions for installation and use of the software on a provided test data set, tells users exactly how to get a result with a small number of explicit steps on a specified test data set.
Informative figures	A schema that explains how the software works, and its modules.
News	Changes in behavior, bug fixes, new features and caveats.
FAQ	Answers to commonly asked or anticipated questions.
Issue tracker	News and discussion of details not otherwise provided in the documentation or not apparent to users. A chanel where users can send questions and feedback, ex: GitHub Issues.
Built-in help	Concise description of a software component and its parameters.
Score	A value between 0 and 1, which is the sum of each feature value divided by the number of features.

CAPÍTULO 3

Artigo 2 - HamleTE: a deep learning-powered tool to annotate transposable elements.

Tiago M. F. F. Gomes, Alexandre R. Paschoal & Elgion L. S. Loreto.

Manuscrito em preparação.

HamleTE: a deep learning-powered tool to annotate transposable elements

Tiago M. F. F. Gomes, Alexandre R. Paschoal, Elgion L. da S. Loreto.

ABSTRACT

Transposable elements (TEs) are DNA sequences capable of changing their location in genomes, a process known as transposition. The repetitive and fragmented nature of TEs makes them difficult to find and classify. To date, we are not aware of any deep learning-based tools that are capable of identifying and classifying TE elements from genomes without the need of a plethora of external tools. HamleTE is a deep learning powered tool that uses a workflow to generate a library to annotate TE from genomes. It uses convolutional neural networks for TEs classification to the level of superfamily. HamleTE works as a classifier and an annotation tool of easy installation and use. Its classification power equals and even surpasses existing classification programs in several aspects, helping to reinforce deep learning methods as another ally in the search for TE in eukaryotic genomes in general.

keywords: Deep learning, transposable elements, annotation, classification, software.

INTRODUCTION

Transposable elements (TE) are DNA sequences able to change their location in genomes, a process known as transposition. Initially, TEs were known as “junk DNA”, due to their highly repetitive nature and unknown roles on genomes. However, research has shown that TE may play various roles on a host genome such as gene expression regulation (Bourque et al. 2018; Drongitis et al. 2019), aging and cancer (Burns 2017; Andrenacci et al. 2020), as well as speciation (Serrato-Capuchina and Matute 2018) and the development of adaptive immunity in vertebrates (Hayward and Gilbert 2022). They can transpose either via an RNA or a DNA intermediate, which is used to classify them into classes. Other structural aspects allow their classification in lower levels as order, superfamily and family (Wicker et al. 2007). TEs that transpose with an RNA intermediate are classed as class I elements and can be further differentiated into LTR elements,

which have a long terminal repeat (LTR) on their structure, or non-LTR elements (Kapitonov et al. 2009; Zhang et al. 2014). Class II elements use a DNA intermediate and are very distinctive from one another, however they can be classified as having or not having a terminal inverted repeat (TIR) (Skipper et al. 2013). The repetitive and fragmented nature of TEs makes them difficult to find and classify in genomes. Additionally, TEs may have undergone a substantial number of mutations and changes over time, so the divergence between sequences of the same order or superfamily, or even family, creates another issue in correctly classifying them (Sotero-Caio et al. 2017; Wells and Feschotte 2020).

Many techniques, such as similarity-based, structure-based, and *de novo* methodologies, have been developed throughout the years to aid in such endeavors (Permal et al. 2012; Goerner-Potvin and Bourque 2018; Storer et al. 2022). The recent rise in popularity of machine learning and deep learning methods brought the attention of bioinformaticians to the use of deep learning in the field of omics sciences (Zou et al. 2019; Martorell-Marugán et al. 2019). The ability of deep learning methods to learn from data without needing to be explicitly programmed according to user defined rules is an advantage in the era of big data, with more and more data being generated (Zhang et al. 2018; Li et al. 2019). Convolutional neural networks (CNN) are one the most used deep learning algorithms for image classification (Rawat and Wang 2017), also being used for other tasks such as computer vision (Luo et al. 2018), natural language processing (Wang and Gang 2018), recommendation systems (Xu et al. 2019), speech recognition (Han et al. 2020), among many others. The convolutional layers of a CNN are used for feature extraction, usually followed by a dimensionality reduction layer called pooling layer. Essentially, the convolutional layer employs sliding-window filters (also known as kernels) to extract features from a dataset, which are subsequently processed by a pooling layer to reduce the number of features, focusing on the most important elements of the data. The information generated from feature extraction using convolutional layers is then sent to the fully-connected layers, which learn the association of the data and their labels in order to correctly categorize data (Rawat and Wang 2017).

In the omics sciences, deep learning has been used, for example, to identify and predict enhancers and promoter regions on the human genome (Oubounyt et al. 2019; Umarov et al. 2019), for prognosis based on transcriptomic data (Ching et al. 2018), identify protein folding (Jumper et al. 2021) and predict gene expression (Zrimec et al. 2020; Avsec et al. 2021). There are tools such as TERL (da Cruz et al. 2021) and DeepTE (Yan et al. 2020) for the classification of TEs using deep learning, and more specifically CNN. However, the latter tools are classifiers only. TransposonUltimate (Riehl et al. 2022) is a pipeline of various tools for TEs annotation that applies different machine learning methods for classification, not using deep neural networks. To date, we are not aware of any deep learning-based tools that are capable of identifying and classifying TE of a genome from start to end without the need of a plethora of external tools. Thus, we present HamleTE, a deep learning powered tool that uses a workflow to annotate TE elements from genomes. HamleTE can be used as an annotation tool using the genome mode and a genome as input or also only as a classifier using the classifier mode, as a way to help curate existing annotated sequences.

MATERIALS AND METHODS

Datasets

The sequences used to construct the TE datasets were obtained from the Conifer transposable elements database (ConTEdb, 322,705 sequences), the Dioecious Plants Transposable Elements Database (DPTEdb, 31,340 sequences), the Salicaceous Plants Transposable Elements Database (SPTEdb, 18,413), the last publicly available Repbase database (2018 version, 55,892 sequences), and sequences) and Soybase transposable elements database (SoyTEdb, 38,664 sequences), totaling 467,014 sequences. Sequences were filtered to remove misclassified and duplicate sequences from the dataset, then, classified into class, subclass, order, and superfamily according to Wicker (2007), resulting in 435,883 sequences.

The non-TE dataset was built of protein coding sequences and non-coding RNA sequences from *Homo sapiens*, *Mus musculus*, *Danio rerio*, *Populus trichocarpa*, *Arabidopsis thaliana*, *Drosophila melanogaster*, *Zea mays* and

Caenorhabditis elegans, with total of 337,014 sequences (239,456 coding, 97,558 non-coding).

To train and test a deep learning model, a dataset is divided into training and validation datasets equally stratified by label; the latter is used to evaluate model performance on data that was not seen on training. We divided the datasets into training and validation using the module *train_test_split* module from the scikit-learn python library. We represented sequences as vectors with the index '1' for 'A', '2' for 'T', '3' for 'G', '4' for 'C' and '5' for 'N'. Any other IUPAC nucleotide representations were replaced by 'N'. All datasets had sequence lengths ranging from 50 to 30,000 nucleotides, which were padded with zeros after transforming sequences to vectors to make all sequences the same length of 30,000. Table 1 shows a summarization of the dataset used for training and testing each model.

The first dataset was used to train a model to differentiate TE from non-TE sequences, and was composed of 18,914 sequences for TE, protein coding sequences and non-coding RNA sequences for each label, totaling 56,742 sequences, 80% of which are for training and 20% for validation. The second dataset, to identify class I and class II TE, had 19,064 sequences in total (9,532 for each class), 80% training and 20% validation. The dataset used for LTR/non-LTR identification was composed of 23,175 LTR sequences and 6,814 non-LTR sequences, 90% for training and 10% for validation. Given that it was a unbalanced dataset label-wise, we used python's scikit-learn *class_weight* module to attribute an initial weight for each label, which was, approximately, 2.20055 for the non-LTR label and 0.64701 for the LTR label. The dataset used to train the model for non-LTR classification was also unbalanced, so we also used the module *class_weights* to give initial weights for each label. The sequences and weights were distributed as follows: *L1*, 25,525 sequences, initial weight of 0.17228; LINE, 1,622 sequences, initial weight 2.71006; *CRE*, 921 sequences, initial weight 4.77276; *DIRS* 779 sequences, initial weight 5.64276; SINE 741, initial weight 5.93213; *RTE*, 698 sequences, initial weight 6.29758; *Penelope*, 494 sequences, initial weight 8.89820. The training and validation split was 90% and 10%, respectively. The rationale for using unbalanced datasets with precomputed weights is to overcome limitations caused by a lack of data for many

superfamilies/labels, allowing for a broader range of classification and avoiding, or at least significantly reducing, overfitting for overrepresented labels, making sense with real world data. The dataset used to classify LTR superfamilies included 11,358 LTR sequences, with 3,786 for *Bel-Pao*, *Copia*, and *Gypsy*, 90% for training and 10% for validation. For Class 2 TE superfamily classification, the dataset was 2,865 for *hAT*, *Helitron*, *Mutator*, *Pif-Harbinger* and *Tc1-Mariner*, 90% for training and 10% for validation, totaling 14,325.

Repeat extraction and clustering

HamleTE extracts and clusters repetitive sequences from genomes or transcriptomes for later classification. We use Red (Girgis 2015) to detect repeats, with a default k-mer size of 13, as recommended by Red's guidelines. A command-line option allows the user to change this value. All other Red parameters, such as the minimum number of occurrences of the k-mer, are set to their default.

To reduce redundancy and retrieve more intact repeats, repeats are clustered using cd-hit-est (Li and Godzik 2006; Fu et al. 2012). The alignment coverage is set to 0.8 (80%), and the cd-hit-est '-G' flag is set to 0 to use a local sequence alignment strategy. Other sequence alignment parameters are set to the cd-hit-est default.

Deep learning models

To identify and classify transposable elements, six models were developed. Tensorflow (Abadi et al. 2016) and Keras (Chollet 2015) frameworks were used to create all models. Model 1 distinguishes TE from non-TE. Model 2 categorizes TE as either class I or class II elements. Model 3 distinguishes between LTR and non-LTR for elements classified as class I by model 2. Model 4 assigns elements classified by model 2 as class II to superfamilies. Models 5 and 6 classify LTRs and non-LTRs at the superfamily level. Supplementary table 1 shows the hyperparameters for each model.

We used the Python package Talos (Autonomio 2020), which automates hyperparameter tuning and model evaluation, to choose the hyperparameter

values for the models. Then, based on the lowest validation loss, we chose the best Talos results and manually tested the hyperparameters to fine-tune them more.

The basic architecture of each model (Figure 1) consists of an input layer, an embedding layer, three one-dimensional convolutional layers, each followed by a pooling layer. Subsequently, there is a normalization layer and a data flattening step. The last layers are two dense layers with a dropout layer among them, with the last dense layer as the output layer. The *Embedding* layers map the input to a dense vector representation of words or characters and establish a semantic relationship between the values, resulting in word embeddings that function as a lookup table, whereas a one-hot encoded representation produces a sparse vector with no relationship between the input values and has a less efficient use of space. In many cases, embedding vectors can help reduce the “curse of dimensionality” (Bauer and Kohler 2019; Chattopadhyay and Lu 2019) when compared to methods like one-hot encoding for data representation. Examples of methods using embedding layers are Word2Vec (Mikolov et al. 2013a) and GloVe (Pennington et al. 2014). *Conv1D* was the chosen convolutional layer, which is the main piece of a convolutional neural network, responsible to detect features of the input by applying a matrix of weights (convolution kernel) producing a resulting vector called feature map. The activation function used was ReLU for all *Conv1D* layers in all models, the number of filters and kernel values for each layer in each model are shown in Table 1. The *MaxPooling* layer applies a sliding window over the feature map values, extracting the highest value and reducing the dimension of the feature map; the pooling size was 7 in all models. The *LayerNormalization* normalizes the input values in all neurons of a layer for each sample. The *Flatten* layer flattens a multi-dimensional tensor to a single dimension. *Dense* layers (or fully connected layers) have the job of matching the learned features to the given labels. The last dense layer must have the number of neurons corresponding to the given number of labels. Table 1 shows the number of filters for each dense layer in every model. For all models, we used categorical crossentropy and adam as the loss function and optimizer, respectively, and 15 as the number of training epochs.

Benchmarks

To assess HamleTE's performance in TE identification and classification, we compared it to DeepTE and TERL, using the EDTA dataset (available on https://github.com/oushujun/EDTA/tree/master/database) and sequences of the repbase dataset that we were able to identify at the levels of class, subclass, order and superfamily. The EDTA dataset contained 3,793 sequences, while the screened Repbase dataset contained 39,996 sequences. Accuracy, specificity, precision, recall and F1-score were the performance metrics used for comparison. *Accuracy* is the correct predicted fraction over all labels. *Specificity*, or true negative rate, shows how well the program can exclude the true negatives for a given label. *Precision* (positive predictive value), denotes the proportion of a positive predicted label being a true positive. *Recall* (sensitivity or true positive rate), shows the ability of the program to correctly predict a true label. *F1-score* is the harmonic mean between precision and recall, and can be understood as a way of measuring a model's accuracy that does not require an estimate of true negatives, being commonly used to compare two or more classifiers. All the metrics are calculated using the formulas on Supplementary figure 1.

We also compared HamleTE against EDTA and RepeatMasker in order to evaluate its classification performance and the generated TE library. EDTA parameters were *--anno 1*, to perform whole-genome TE annotation, and *--force 1*, to not interrupt and exit when no confident TE candidates are found. RepeatMasker parameters were *-no_is* (skips IS element search), *-nolow* (skips low complexity masking) and *-norna* (skips small RNAs masking). HamleTE was run using the default configuration. We used the previously described datasets (EDTA and Repbase) for the classification task, and the genomes of *Drosophila melanogaster* (version refseq release 6.32), *Drosophila simulans* (release 2.02), *Drosophila virilis* (release 1.07), *Danio rerio* (GRCz11), *Cicer arietinum* (ASM33114v1), *Oryza sativa* (IRGSP-1.0) and *Zea mays* (GCA_902167145.1) were used to evaluate the generated TE library of HamleTE, RepeatMasker and EDTA.

RESULTS

Model training and testing

Model 1 had a training accuracy of around 96% and a validation accuracy of 0.894 in identifying TE and non-TE (Supplementary Figure 2). The TE accuracy was 0.95 and f1-score 0.927 for the validation dataset (Figure 2 Model 1).

Model 2 training accuracy was 98.8%, while validation accuracy was 0.945 (Supplementary Figure 3). For the validation dataset, the Retrotransposon accuracy was 0.945 and f1-score of 0.946; for DNA transposon identification accuracy was 0.945 and f1-score of 0.943 (Figure 2 Model 2).

Model 3, for the LTR/non-LTR identification task, had a training accuracy of 0.97 and validation accuracy was, approximately, 0.95 (Supplementary Figure 4). For the LTR TE validation data, accuracy was 0.95 and f1-score 0.972; for non-LTR validation data, accuracy was around 0.95 and f1-score 0.906 (Figure 2 Model 3).

Model 4 training accuracy was 0.95 with a validation accuracy of 0.894 for classifying DNA transposon superfamilies (Supplementary Figure 5). Regarding the validation data, the superfamily classification mean accuracy was 0.958, mean specificity 0.974 and mean f1-score of 0.895. The *Mutator* element had the highest accuracy (0.968) and f1-score (0.921); *Helitron* had the highest recall, 0.955. The *hAT* element had the lowest f1-score, 0.864. The complete results are shown on Figure 2 Model 4.

Model 5, for classifying LTR superfamilies, training accuracy was 0.93, while the validation accuracy was 0.853 (Supplementary figure 6). Validation data-wise, the superfamily classification mean accuracy was 0.902, mean specificity 0.926 and f1-score of 0.852. The *Bel-Pao* element had the best results overall, accuracy 0.945 and f1-score of 0.92 (Figure 2 Model 5).

Model 6, the non-LTR superfamily classifier, had a training accuracy of 0.916 and a validation accuracy of 0.886 (Supplementary figure 7). The superfamily classification mean accuracy was 0.967, mean specificity 0.981 and mean f1-score of 0.69, in relation to the validation data. The *L1* element had an accuracy of 0.925 and the highest f1-score, 0.952; for SINE, accuracy was 0.993 and f1-score 0.859. The complete results are shown on Figure 2 Model 6.

TE identification and classification

From a total of 39,996 TEs in the filtered repbase dataset, HamleTE identified 38,369 as TEs, TERL 38,971, DeepTE 33,581, EDTA 2,553 and RepeatMasker 9,818 (Figure 3a). Out of 3,792 TEs in the EDTA dataset, HamleTE identified 3,224 as TEs, TERL 3,576, DeepTE 1,906, EDTA 1,451 and RepeatMasker 3,675 (Figure 3b).

In regards to the repbase dataset prediction metrics (Figure 4a), the mean accuracy and f1-score for HamleTE were 0.963 (± 0.039) and 0.665 (± 0.18), respectively; for TERL, accuracy was 0.92 (± 0.043) and f1-score 0.658 (± 0.09); for DeepTE, accuracy was 0.955 (± 0.05) and f1-score 0.621 (± 0.277); EDTA mean accuracy and f1-score were 0.745 (± 0.376) and 0.189 (± 0.221), respectively; for RepeatMasker values were 0.951 (± 0.069) and 0.36 (± 0.287) for mean accuracy, and f1-score respectively.

Label-wise (Figure 5a), HamleTE has correctly predicted 1,800 *Bel-Pao* elements out of 1,875, f1-score 0767; TERL correctly classified 1,578 elements, f1-score 0.628. Neither DeepTE, nor EDTA nor RepeatMasker were able to correctly classify *Bel-Pao* elements. For the *Copia* element, HamleTE classified 6,080 out of 7,101, f1-score 0.798; DeepTE classified 5,603 elements, f1-score 0.862; TERL correctly classified 5,993 elements, f1-score 0.746; EDTA identified 949 elements, f1-score 0.235; RepeatMasker identified no *Copia* elements. HamleTE *Gypsy* correct predictions were 8,594 out of 11,239, f1-score 0.764; DeepTE identified 8,085 elements, f1-score 0.728; TERL identified 6,677 elements; EDTA identified 970 elements, f1-score 0.158; RepeatMasker identified 1,156 elements, f1-score 0.181. For the *Helitron* element, HamleTE correct predictions were 906 out of 976, f1-score 0.552; EDTA identified 387 elements, f1-score 0.541; RepeatMasker identified 137 elements, f1-score 0.245. Neither DeepTE nor TERL could identify *Helitron* elements. Out of 1,744 *L1* elements, 633 were classified as *L1* and 837 as *LINE* by HamleTE, f1-score 0.519; TERL correctly classified 1,676, f1-score 0.508; RepeatMasker identified 1,071 *L1* elements, f1-score 0.649. Neither DeepTE nor EDTA identified any *L1* elements. HamleTE *hAT* correct predictions were 2,827 out of 3,111, f1-score 0.89; DeepTE

correct classified 2,626 elements, f1-score 0.702; TERL classified correctly 2,844 elements, f1-score 0.781; EDTA identified correctly only 3 elements; RepeatMasker identified 1,342 elements, f1-score 0.529. HamleTE *Tc1-Mariner* correct predictions were 2,218 out of 2,632, f1-score 0.86; DeepTE identified 2,219 *Tc1-Mariner* elements, f1-score 0.728; TERL identified 2,465 elements, f1-score 0.657; RepeatMasker identified 856 elements, f1-score 0.45. EDTA did not identify any *Tc1-Mariner* elements. All counts per label are available on supplementary table 2-6.

The EDTA dataset prediction (Figure 4b) mean accuracy and f1-score for HamleTE were 0.945 (± 0.037) and 0.504 (± 0.2), respectively; for TERL, accuracy was 0.882 (± 0.046) and f1-score 0.475 (± 0.281); for DeepTE, accuracy was 0.938 (± 0.036) and f1-score 0.430 (± 0.284); EDTA mean accuracy and f1-score were 0.932 (± 0.041) and 0.378 (± 0.288), respectively; for RepeatMasker values were 0.897 (± 0.078) and 0.127 (± 0.056) for mean accuracy and f1-score respectively.

On a per-label basis (Figure 5b), HamleTE correctly classified 255 out of 292 *Helitron* elements, f1-score 0.667; RepeatMasker identified only 16 *Helitron* elements, f1-score 0.104. TERL and DeepTE did not identify any *Helitron* elements. For the *Tc1-Mariner*, HamleTE identified 32 out of 47 elements, f1-score 0.372; DeepTE identified 17 elements, f1-score 0.309; TERL identified 46 elements, f1-score 0.12; EDTA identified only 4 elements. RepeatMasker did not identify *Tc1-Mariner* elements. For the *hAT* element, HamleTE correct identified 457 out of 592, f1-score 0.848; DeepTE identified 182 elements, f1-score 0.444; TERL identified 524 elements, f1-score 0.714; EDTA identified 87 elements, f1-score 0.255; RepeatMasker identified 21 elements, f1-score 0.068. For *Copia*, 340 out of 514 were correctly identified by HamleTE, f1-score 0.658; DeepTE 182 elements, f1-score 0.84; TERL correctly classified 383 elements, f1-score 0.655; EDTA identified 376 elements, f1-score 0.807. RepeatMasker did not identify any *Copia* elements. HamleTE correctly identified 527 out of 734 *Gypsy* elements, f1-score 0.705; DeepTE identified 534 elements, f1-score 0.788; TERL identified 539 elements, f1-score 0.665; EDTA identified 484 elements, f1-score 0.77;

RepeatMasker identified 55 elements, f1-score 0.138. All counts per label are available on supplementary tables 7-11.

Generated TE libraries

In the first step of HamleTE's workflow (Figure 6) for the *D. melanogaster* genome, Red extracted 65,503 repeat sequences, which were then clustered by cd-hit reducing the total amount to 59,534 sequences from which 8,731 (14.66 %) were identified as TEs (supplementary figure 8). The most represented category was LTR with 2,779 sequences, followed by TIR 2,679, non-LTR 1,790 and Helitron 1,483 (Figure 7). From the total identified TEs, the most represented superfamilies classified by HamleTE were *Helitron* (16.98 %), *Copia* (14.63 %), *Gypsy* (12.43 %), *Tc1-Mariner* (12.33 %), *Mutator* (8.19 %) and *hAT* (7.58 %) (Supplementary figure 9). RepeatMasker generated a TE library of 1,124 sequences in which the most represented category was non-LTR (745 sequences), TIR (186), LTR (162), Helitron (2) and 29 sequences of Unknown order (Figure 8); the most represented superfamilies/families were *CR1* (20.11 %), *hAT-Ac* (10.14 %), *L2* (10.05 %), the SINE element *5s-Deu-L2* (9.34 %) and *Gypsy* (8.9%) (Supplementary figure 9). EDTA generated a TE library of 598 sequences. The most represented categories respectively were TIR (312 sequences), LTR (222) and Helitron (64) (Figure 9); *Gypsy* (30.10 %) was the most represented element, followed by DTC (*CACTA*, 19.9 %), DTM (*Mutator*, 16.55 %), *Helitron* (10.7 %) and a MITE-DTC element (4. 85 %). Full results for each software and classes are shown in figure Supplementary figure 9.

For the *O. sativa* genome, by using Red, HamleTE found 247,682 repeats, clustered by cd-hit in 205,211 sequences of which 73,280 (35.71 %) were classified as TEs (supplementary figure 8). The number of TIR sequences was 28,782, LTR 21,041, non-LTR 13661 and Helitron 9,796 (Figure 7). For superfamilies, *Helitron* represented 13.36 % of the total, *Tc1-Mariner* 13.25 %, *Gypsy* 13.11 %, *Copia* 11.8 %, *Mutator* 9.3 %, *hAT* 9.11 % and *Pif-Harbinger* 7.61 % (Supplementary figure 10). RepeatMasker TE library generated a total of 2,882 sequences, with the most represented category being non-LTR (1,590 sequences), LTR (757), TIR (419), Helitron (102) and 14 sequences of Unknown

order (Figure 8). Superfamilies were composed mainly of *Gypsy* (13.84 %), *CR1* (12.18 %), *ERVL* (9.02 %), *L2* (8.85 %), the SINE element *5s-Deu-L2* (8.53 %), *RTE-BovB* (7.6 %) and *hAT-Ac* (6.7%) (Supplementary figure 10). The total sequences of the EDTA library for *O. sativa* was 7,183 of which 5,273 were from the category TIR, 1,277 LTR and 633 Helitron (Figure 9). The most represented superfamilies were DTT (*Tc1-Mariner*) with 34.3 % of the total TE library, *Gypsy* 10.06 %, DTM (*Mutator*) 9.62 %, MITE-DTT 9.59 %, *Helitron* 8.81 % and DTC (*CACTA*) 7.92 % (Supplementary figure 10).

HamleTE repeat identification step for the *C. arietinum* genome resulted in 222,234 repeats, clustered in 162,049 sequences. From that, HamleTE classified 72,544 (44.8 %) as TE (supplementary figure 8). The number of TIR sequences were 26,470, LTR were 17,987, non-LTR 16,553 and Helitron 11,534 (Figure 7). The most represented superfamilies were *Mutator* (17.36 %), *Helitron* (15.89 %), *Copia* (13.79 %), *Tc1-Mariner* (11 %), *Gypsy* (7.81 %) and *hAT* (6.15 %) (Supplementary figure 11). RepeatMasker resulted in a library of 2,864 sequences of which 1,770 were non-LTR, 516 LTR, 440 TIR, 97 Helitron and 41 of unknown order (Figure 8). Superfamilies were composed mostly by *CR1* (17.5 %), *L2* (10.1 %), *5s-Deu-L2* (9.39 %), *L1* (7.82 %), *Gypsy* (7.54 %), *ERLV* (6.98 %) and *hAT-Ac* (6.91 %) (Supplementary figure 11). EDTA generated a library of 2,201 sequences of which 1,141 were classified as LTR, 916 as TIR and 144 as Helitron, regarding the TE order (Figure 9). For superfamilies, *Copia* (32.80 %) was the most represented, followed by DTM (*Mutator*, 12.58 %), Unknown LTRs (9.68 %), *Gypsy* (9.36 %), MITE-DTM (8.86 %), DTC (*CACTA*, 8.54 %) and *Helitron* (6.54 %) (Supplementary figure 11).

In the genome of *D. rerio*, the 1,717,574 repeats found were clustered in 1,285,892 sequences of which 385,547 (29.99 %) were identified as TEs by HamleTE (supplementary figure 8). Order-wise, TIR was the most representative with 124,040 sequences, followed by LTR with 117,898, non-LTR 82,532 and Helitron 61,068 (Figure 7). The most represented superfamilies were *Helitron* (15.83 %), *Copia* (14.52 %), *Tc1-Mariner* (10.65 %), *Gypsy* (9.44 %), *hAT* (9.34 %), *Mutator* (8.5 %) and *Bel-Pao* (6.62 %) (Supplementary figure 12). For RepeatMasker, from a library of 255,162 sequences, the number of TIR

sequences was 161,370, non-LTR 84,677, LTR 8,652 and 463 of unknown Order (Figure 8). For superfamilies, *Tc-Mar-Tigger* represented 58.15 % of the total TE library, *L2* 12.54 %, *5s-Deu-L2* 10.78 %, *hAT-Charlie* 4.36 % and *L1* 4.02 % (Supplementary figure 12). In the EDTA library of 10,949 sequences, the number of TIR sequences was 7,490, LTR 3,352 and Helitron 107, regarding the Order (Figure 9). Superfamilies were mostly represented by DTA (*hAT*, 25.36 %), *Gypsy* (23.25 %), DTC (14.6 %), MITE-DTA (10.05 %) and DTM (*Mutator*, 7.38 %) (Supplementary figure 12).

The number of repeats found in HamleTE's first step in the *D. simulans* genome was 188,343, reduced to 183,336 after clustering, of which 44,167 (24.09 %) were classified as TEs (supplementary figure 8). The most represented category was TIR with 15,985 sequences, followed by LTR 15,447 sequences, Helitron with 6783, non-LTR with 5952 (Figure 7). The *Helitron* superfamily represents 15.35 % of the total TE library, *Copia* 14.38 %, *Tc1-Mariner* 11.25%, *hAT* 10.77%, *Bel-Pao* 10.38 %, *Gypsy* 10.22 % and *Mutator* 8.85 % (Supplementary figure 13). RepeatMasker generated a library of 828 sequences of which 532 were non-LTR, 172 TIR, 89 LTR, 31 of unknown TE order and 4 Helitron (Figure 8). For superfamilies/families, *CR1* (15.22%) was the most represented, followed by *L2* (14.85 %), *hAT-Ac* (12.44 %), *RTE-BovB* (10.99 %), the SINE element *MIR* (8.21 %) and *Gypsy* (4.95 %) (Supplementary figure 13). From the total of 379 sequences of the EDTA library generated from *D. simulans* genome, 298 were from the Order TIR, 69 Helitron and 12 LTR (Figure 9). DTC was 32.72 % of the total, DTM 22 %, *Helitron* 18.20 %, MITE-DTA 6.6 % and MITE-DTC 6.33 % (Supplementary figure 13).

In the *D. virilis* genome, HamleTE found 124,751 repeats, clustered in 103,367 sequences of which 19.064 (18.44 %) were identified as TEs (supplementary figure 8). The most represented category was LTR (6,616 sequences), followed by TIR (5,099), non-LTR (3,907) and Helitron (3,442) (Figure 7). The superfamily *Helitron* represented 18 % of the total library, *Copia* 17.34 %, *Gypsy* 11.35 %, *Tc1-Mariner* 10.08 %, *Mutator* 7.9 %, *hAT* 6.23 % and *Bel-Pao* 6% approximately (Supplementary figure 14). RepeatMasker generated a library of 1,862 sequences, of which 1,070 were non-LTR, 451 LTR, 327 TIR, 12 of

unknown order and 2 Helitron (Figure 8). Superfamilies/families were composed mainly by *CR1* (21.7 %), *ERV1* (16.92 %), *hAT-Ac* (11.55 %), *L2* (10.52 %), *L1* (7.78 %) and *ERVL* (5.48 %) (Supplementary figure 14). EDTA generated a library of 588 sequences, with the most represented TE order being TIR (355 sequences), then LTR (133) and Helitron (100) (Figure 9). The most representative elements were DTM (28.23 %), *Gypsy* (19.56 %), *Helitron* (17 %), MITE-DTM (11.22 %) and DTC (9.86 %) (Supplementary figure 14).

The number of repeats found in the *Z. may* genome running HamleTE was 902,671, clustered into 494,401 sequences (supplementary figure 8). From this total, 177,935 (36 %) were classified as TEs. LTR was the most represented type with 61,326 sequences, followed by non-LTR with 41,676 (Figure 7), Helitron with 39,090 and TIR with 35,843. The most representative superfamilies were *Helitron* (21.96 %), *Gypsy* (19.07 %), *Copia* (12.19 %), *hAT* (6.01 %) and *Mutator* (5.93 %) (Supplementary figure 15). RepeatMasker generated a library of 17,244 sequences formed by 7,939 non-LTR sequences, 5,497 LTR, 3,231 TIR, 431 Helitron and 146 of unknown order (Figure 8). For superfamilies/families, the most represented was *Gypsy* (27.38 %), then *CR1* (12 %), *hAT-Ac* (11.02 %), *L1* (9.68 %), *RTE-BovB* (7.17 %) and *L2* (6.23 %) (Supplementary figure 15). The EDTA library of 29,301 sequences was represented by 18,057 elements of the LTR Order, 7,880 for TIR and 3,364 for Helitron (Figure 9). *Gypsy* (31.23 %) was the most represented element followed by unknown LTR elements (18.64 %), *Copia* (11.75 %), *Helitron* (11.48 %), DTC (6.45 %) and DTA (5.67 %) (Supplementary figure 15).

DISCUSSION

The annotation and classification of TEs in a computational way is a long-standing challenge in which several strategies have been employed to try to solve the problem (Ou et al. 2019; Bell et al. 2022). Annotation can be done manually or automatically. Manual annotation is the most accurate, achieving deeper levels of classification such as, for example, subfamilies (Carey et al. 2021). However, it requires a specialist in the field to obtain more refined results, it is a time-consuming task, practically unfeasible on a large scale, and even so, it

requires, at first, the help of automatic methods of annotating TEs (Arkhipova 2017; Goubert et al. 2022).

Software for automatic annotation of TEs, in general, use methods based on similarity, motif search, *de novo* methods or a combination of two or more of these. Some of the main examples of software used to annotate TEs from genomes are RepeatMasker, RepeatModeler, REPET and EDTA (Goerner-Potvin and Bourque 2018; Rodriguez and Makałowski 2022). Traditional automatic annotation methods tend to be computationally expensive and, in many cases, the software used is not user-friendly, requiring pre-processing that may not be trivial, especially by users not specialized in bioinformatics and programming (Ison et al. 2020; Krampis 2022).

New tools implementing the use of deep learning (DL) have had a significant increase in their number in several areas of scientific research in recent years, as in the case of bioinformatics (Peng et al. 2018; Wang et al. 2018). With regard to TEs, tools such as TERL and DeepTE make use of DL to classify TEs, however, they do not propose a workflow for the annotation of transposable elements from genomes. HamleTE seeks to fill this gap by creating a simple and efficient workflow for identifying and classifying repetitive sequences, powered by the use of DL, more specifically, convolutional neural networks (CNN).

HamleTE has an annotation mode, returning as its output the repeats classified as TEs and their coordinates in the genome, and a classification mode, allowing its standalone use as a classifier, or to help refine or support classification results of other tools. Its performance equals or exceeds that of the primary tools used for annotation and classification of TEs in many circumstances, and is user-friendly in both its installation and usage, a important topic regarding software adhesion for a non-technical use specially (Lawlor and Sleator 2020; Baril et al. 2022). HamleTE was designed to also be used on personal computers without causing overhead (*i.e.*, excessive processing or memory usage) while maintaining adequate performance at all times.

Its installation can be done using a virtual environment with a conda recipe or a Docker container, in addition to allowing manual installation for more advanced users. HamleTE splits files in batches, with the option to set this value,

making it possible to control the amount of memory to be used, which avoids exceeding limits causing computer freezes on machines with fewer resources, allying performance without sub-optimal performance (Cirillo et al. 2021). HamleTE also has options for configuring *cutoff* values for classifying transposable elements. A value can be established to filter the classification of repeats as being TEs and another value to mask the classification at the superfamily level, resulting in a more reliable final classification. HamleTE also has the option to set the size of the k-mers used to search for repetitive sequences in a genome when in annotation mode, which is a further aid in the refinement of the final annotation. Being able to choose the k-mer size is very important since one should always take into consideration the size of the genome to choose the *k* value, which has an impact on the results (Chikhi and Medvedev 2014; Contreras-Moreira et al. 2021).

The repetitive nature of TEs creates the need for tools capable of accurately identifying them in genomes, which is not a trivial task (Wells and Feschotte 2020). The annotation programs currently available, such as RepeatMasker, use other programs in their pipelines such as RepeatScout, Recon and GRF. Despite being well established, these tools tend to be quite slow or have a high rate of false positives (Flutre et al. 2011; Girgis 2015). In the first part of its workflow in annotation mode, HamleTE identifies repetitive sequences with the help of the Red program, a *de novo* tool for masking repeats with a very low false positive rate and high speed, surpassing the best known programs (Contreras-Moreira et al. 2021). The output of the Red program returns small repetitive sequences in tandem, low complexity and redundant sequences. To solve this aspect, HamleTE filters the repeats by size and then uses the “cd-hit-est” program to cluster the sequences, reducing the existing redundancies, resulting in less artifacts, less computation time and an improved TE library (Ono et al. 2015; Ahsan et al. 2023). There is an option to skip the clustering step for users interested in doing further in-depth evaluation of the sorted sequences later.

The high variability between the elements of different TE classification categories, and even between TEs of the same hierarchical level of classification, makes it difficult for any model to learn the parameters of a given category because it introduces great complexity to the data, preventing the model from

generalizing the features of the data (Cui et al. 2019; Shahinfar et al. 2020). Similarly, the more categories the model must learn to categorize, the more difficult learning gets, preventing the model from establishing a categorization decision threshold between labels (Abramovich and Pensky 2019). This categorization learning difficulty might also emerge owing to a lack of training data. Generalization failure can cause the model to overfit, which is when the model learns too well only on the training data and is unable to adequately evaluate data outside of those presented during training, whereas learning failure training data results in the process of underfitting (Bashir et al. 2020). HamleTE employs six different classification models that follow a hierarchical workflow, as shown in Figure 6, beginning with the distinction of TE candidate, coding genes and non-coding mRNA sequences, progressing through the steps of identifying TE class, order up to superfamilies, with the goal of reducing the level of complexity and variance per step, thereby reducing the possibility of overfitting or underfitting (Ashiquzzaman et al. 2018; Gavrilov et al. 2018).

Lack of data for a given category relative to another can cause overfitting/underfitting of the different categories within the same model. Having an equal number sequences for all the labels to be classified tends to avoid this problem, however, the data available in the real world can differ greatly in number for the different classes (Saini and Susan 2022). Among the different ways to get around this are undersampling, the selection of a sample from the category with the highest representativeness in equal numbers to the category with the lowest representativeness (Hernandez et al. 2013). The problem with this approach is that the category that has been undersampled may not contain a good representation of its features depending on the amount of reduction done (Johnson and Khoshgoftaar 2019). Another way to approach this problem is to previously establish different weights for each category, creating a higher penalty for errors in categories with fewer sequences (Krawczyk et al. 2014; Liu et al. 2022). This strategy of weighted categories proved to be efficient as can be seen in the case of model 3 for distinguishing sequences classified as Retroelements into LTR and non-LTR (Figure 2c), and model 6 (Figure 2f), for order/superfamily level classification of non-LTR elements, which despite lower metrics in some

cases, showed considerable values for classes with few representations such as SINE, *CRE* and *Penelope* that would otherwise be left out of the training.

In order for CNN models to extract the features of each category of TEs, the sequences need to be represented in numerical form (Zou et al. 2019). Many DL models encode the data into numerical arrays using the one-hot encoding (OHE) method (Supplementary figure 16). This method has the advantage of being fast and simple to implement to represent almost any categorical data. However, it results in a sparse vector (only 0's and 1's), making no effective use of space and establishing no semantic relationship between the data (Rodríguez et al. 2018; LIANG Jie 2019). HamleTE implements in its classification models the transformation of sequences into embedding vectors (Supplementary figure 17), the representation of sequences as dense vectors, which are learned by using an embedding layer prior to the convolutional layers. Embedding layers are used in DL models for natural language processing (Li and Yang 2018; Wang and Gang 2018). The embedding layer learns weights that capture the context of the base order relationship in the sequences and are updated at each training epoch along with the convolutional and dense layers of the model (Mikolov et al. 2013b; Hrinchuk et al. 2020).

The idea of using the sequences as embedding vectors instead of the more commonly used OHE method is because, in a way, the DNA sequences can be seen as sentences of a language, since the order of the bases and structures influence the meaning of the sequence, *i.e.*, the sequence of bases is not random, it gives meaning to the gene product (Searls 1992; Attard et al. 1996; Searls 2002; Wahab et al. 2021; Ji et al. 2021). During the training phase we were able to improve the metrics of the models by replacing the OHE method with embedding vectors (data not shown). The metrics of the HamleTE results compared to other classification programs using DL show that there was no loss in performance when using embedding vectors. Conversely, the use of this approach coupled with fine tuning of the model hyperparameters suggest its success.

In the analyses measuring classification competence with the Repbase and EDTA datasets, HamleTE was second in the number of correct identifications of sequences as TEs (Figure 3a), second only to TERL by a small margin and well

ahead of the other programs. The precision and specificity metrics suggest that TERL tends to generate more false positives in comparison to HamleTE. Compared to DeepTE, HamleTE outperforms it in all metrics for the identification of TEs in the Repbase dataset test (Figure 4a) and falls behind DeepTE in the identification of TEs in the EDTA dataset only slightly in specificity and accuracy, with a higher accuracy, recall and f1-score. When analyzing the metrics of accuracy, specificity, and f1-score, HamleTE is the most balanced of all programs, matching or beating most, and very close to programs using combined TE identification strategies such as RepeatMasker and EDTA when it comes to true negative and true positive rates.

The results of HamleTE in annotation mode with the genomes of 6 species demonstrates congruence with that found in the literature regarding the distribution of TEs in these species. Comparing the distribution of LTR, non-LTR, TIR and Helitron from the annotation of HamleTE, EDTA and RepeatMasker, we notice that EDTA was not able to identify non-LTR TEs in the genomes analyzed, and in the case of the genus Drosophila, which generally has a higher number of LTR elements (Mérel et al. 2020), there is a wide representation of elements of the order TIR in the library generated and a low representation of LTR elements, with an even lower number in the case of *D. simulans* compared to the result for the other two species of the genus Drosophila (Figure 9). The library generated by RepeatMasker showed non-LTR TEs as the most represented type in all genomes (Figure 8), except in the case of *D. rerio* where TIR elements were the most numerous (Meena et al. 2012). Even in the genome of well-studied organisms like *D. melanogaster* and *Z. mays* in which LTR elements are already known to be predominant (Anderson et al. 2019; Stitzer et al. 2021), RepeatMasker generated few elements of this order in its library, especially in the case of *D. melanogaster*. HamleTE was able to identify LTR, non-LTR, TIR and Helitron elements in all genomes, in general, identifying the predominant type in each genome correctly, with the exception of *D. simulans*, in which it identified more elements of the TIR order than LTR, the two most enriched in the generated library. Even so, the number of LTR was quite high, as expected in the species (Petersen et al. 2019; Mombach et al. 2022).

When checking the annotation results at the superfamily level, it is noted that HameleTE identified *Helitron* as the most represented superfamily in number of sequences in 6 of the 7 genomes. Helitrons are TEs that do not possess common TE features such as the presence of terminal repeats, nor do they leave molecular "scars" like TSDs when they are transposed, and when they are transposed by the rolling circle mechanism (also called "peel-and-paste") (Di Stefano 2022), they tend to capture fragments of sequences of protein-coding genes or of other TEs, making them difficult to identify (Han et al. 2019; Hu et al. 2019). Even programs that specialize in identifying Helitrons such as HelitronScanner (Xiong et al. 2014) can encounter difficulties in correctly identifying these elements. In a test done by Ou (2018), of the sequences classified as Helitrons, 21% were LTR elements and 11% TIR elements. In Supplementary table 2 containing the results of the Repbase dataset classification benchmark, among the sequences classified as *Helitron* by HamleTE, approximately 8% were from the *Gypsy* superfamily, 7% *Tc1-Mariner* and 5% *hAT*. However, factors like this do not invalidate the annotation since it is a common event for any automated annotation software.

HamleTE results for TE superfamilies in *D. melanogaster* indicate a high percentage of *Helitron*, *Gypsy* and *Copia* elements in total sequence number. Indeed, Gypsy and Helitron are elements in large proportion in the *D. melanogaster* genome, *Copia* however, is not among the most prevalent (Mérel et al. 2020; Lopik et al. 2023). Observing the distribution of sequences in relation to their size (Supplementary figure 18), there is a considerable amount of *Copia* elements in the 100 to 400 nucleotide range, unlike the more harmonic distribution for *Gypsy* and *Helitron*. Comparing the total bases of *Gypsy* and *Copia* elements in the resulting library (Supplementary figure 19), there is a vast difference, with a preponderance of *Gypsy* elements over *Copia* in the total library fraction, suggesting consonance with what is expected for the *D. melanogaster* genome. The smaller sizes of this large portion of elements classified as *Copia* match the sizes of the long terminal repeats of TEs of the LTR order, on average 359 base pairs, with around 276 for *Copia* elements (Rubin et al. 2011; You et al. 2015). LTRs have regulatory regions required for transcription, including transcription

factor binding sites. There is evidence of their co-option as regulatory regions in different organisms, including humans and mice, which are believed to have contributed in high proportion as binding sites for different transcription factors (Thompson et al. 2016; Deneweth et al. 2022). The presence of this large number of small sequences could mean solo-LTRs (Vitte and Panaud 2003; Jedlicka et al. 2020; Autio et al. 2021), which, in genomes such as *Candida albicans*, are the most represented among the LTR TE (Zhang et al. 2014) families, or TE fragments found in the repeats identification phase of the HamleTE workflow in annotation mode.

The lower number of sequences in the EDTA and RepeatMasker libraries can be explained by taking into account that both programs seek to generate a library of TEs with low redundancy and good reliability, a fact supported by observing the values of the metrics of these programs in the classification benchmark tests, where we observe that they obtain good metrics of specificity and accuracy, (Figure 4a and 4b) despite the poor results in other metrics in general. In addition, the values presented refer to the total number of sequences in the libraries generated by the three software programs and not the representation of the elements in percentage of bases in the genomes, as there is difference between the number of copies of an element within the genome and how much it represents of the total bases. An example is the work of (Stitzer et al, 2021), which showed that there are a close number of copies of the elements *Gypsy*, *Tc1-Mariner* and *Helitron* in the maize genome (76,306, 66,479 and 62,291 respectively), but in total amount of bases, *Gypsy* represents around 776 Mb, while *Tc1-Mariner* and *Helitron* represent 23 Mb and 21 Mb respectively, *i.e.* approximately 35 times more copies of the former.

However, the amount of sequences of the elements for each category, either at the order or superfamily level, is important within the research (Britten 2006; Anderson et al. 2019; Sexton and Han 2019; Walker et al. 2023), as the automatic annotation can be used for refinement of TEs by manual annotation, identification of families and subfamilies (Goubert et al. 2022), or even the use of the library generated in an automated way in programs such as RepeatMasker in the search for elements in newly sequenced genomes (Rodriguez and Arkhipova

2023), and in the search for TEs in closely-related species or to mask TEs in genomes (Rossi et al. 2001; Bell et al. 2022). Tools based on sequence alignment may not be able to identify more distantly related elements of the same superfamily or family without a reference (Pearson 2013; Zielezinski et al. 2017). Reference libraries with marked imbalance, with little or no representation of a given category, such as non-LTRs with EDTA software (Figure 9) or the very low number of LTR elements in *D. melanogaster* and TIR elements in *O. sativa* with RepeatMasker (Figure 8), could have a major impact on the final result of a search relying on automatic annotation based on sequence similarity. In view of this, it is essential to have a library generated by automatic annotation with a balanced representation of TEs, as close as possible to what is expected according to the literature for the species in question, highlighting the importance of having software with alternative approaches helping to dive into eukaryotic genomes in the search for TEs.

CONCLUSION

HamleTE presents a new alternative for the task of TE annotation and classification, helping to reinforce deep learning methods as another ally in the search for TEs in eukaryotic genomes. Through the use of CNNs, its classification power equals and even surpasses existing classification programs in several aspects, also functioning as an annotation tool of easy installation and use, by integrating tools for repeat extraction and redundancy removal in its workflow, resulting in a robust annotation of TEs in eukaryotic genomes in general.

REFERENCES

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al. (2016) Tensorflow: A system for large-scale machine learning. 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). pp 265–283

Abramovich F and Pensky M (2019) Classification with many classes: challenges and pluses.

Ahsan MU, Liu Q, Perdomo JE, Fang L and Wang K (2023) A survey of algorithms for the detection of genomic structural variants from long-read sequencing data. Nat Methods 1–16.

Anderson SN, Stitzer MC, Brohammer AB, Zhou P, Noshay JM, O'Connor CH, Hirsch CD, Ross-Ibarra J, Hirsch CN and Springer NM (2019) Transposable elements contribute to dynamic genome content in maize. Plant J 100:1052–1065.

Andrenacci D, Cavaliere V and Lattanzi G (2020) The role of transposable elements activity in aging and their possible involvement in laminopathic diseases. Ageing Res Rev 57:100995.

Arkhipova IR (2017) Using bioinformatic and phylogenetic approaches to classify transposable elements and understand their complex evolutionary histories. Mob DNA 8:19.

Ashiquzzaman A, Tushar AK, Islam MdR, Shon D, Im K, Park J-H, Lim D-S and Kim J (2018) Reduction of Overfitting in Diabetes Prediction Using Deep Learning Neural Network. In: Kim KJ, Kim H and Baek N (eds) IT Convergence and Security 2017. Springer, Singapore, pp 35–43

Attard GS, Hurworth AC and Jack JP (1996) Language-like features in DNA: transposable element footprints in the genome. Europhys Lett 36:391.

Autio MI, Bin Amin T, Perrin A, Wong JY, Foo RS-Y and Prabhakar S (2021) Transposable elements that have recently been mobile in the human genome. BMC Genomics 22:789.

Autonomio (2020) Autonomio Talos.

Avsec Ž, Agarwal V, Visentin D, Ledsam JR, Grabska-Barwinska A, Taylor KR, Assael Y, Jumper J, Kohli P and Kelley DR (2021) Effective gene expression prediction from sequence by integrating long-range interactions. Nat Methods 18:1196–1203.

Baril T, Imrie RM and Hayward A (2022) Earl Grey: a fully automated user-friendly transposable element annotation and analysis pipeline. doi: 10.21203/rs.3.rs-1812599/v1

Bashir D, Montañez GD, Sehra S, Segura PS and Lauw J (2020) An Information-Theoretic Perspective on Overfitting and Underfitting. In: Gallagher M, Moustafa N and Lakshika E (eds) AI 2020: Advances in Artificial Intelligence. Springer International Publishing, Cham, pp 347–358

Bauer B and Kohler M (2019) On deep learning as a remedy for the curse of dimensionality in nonparametric regression. Ann Stat 47:2261–2285.

Bell EA, Butler CL, Oliveira C, Marburger S, Yant L and Taylor MI (2022) Transposable element annotation in non-model species: The benefits of species-specific repeat libraries using semi-automated EDTA and DeepTE de novo pipelines. Mol Ecol Resour 22:823–833.

Bourque G, Burns KH, Gehring M, Gorbunova V, Seluanov A, Hammell M, Imbeault M, Izsvák Z, Levin HL, Macfarlan TS et al. (2018) Ten things you should know about transposable elements. Genome Biol 19:199.

Britten R (2006) Transposable elements have contributed to thousands of human

proteins. Proc Natl Acad Sci 103:1798–1803.
Burns KH (2017) Transposable elements in cancer. Nat Rev Cancer 17:415–424.
Carey KM, Patterson G and Wheeler TJ (2021) Transposable element subfamily annotation has a reproducibility problem. Mob DNA 12:4.
Chattopadhyay A and Lu T-P (2019) Gene-gene interaction: the curse of dimensionality. Ann Transl Med 7:813.
Chikhi R and Medvedev P (2014) Informed and automated k-mer size selection for genome assembly. Bioinformatics 30:31–37.
Ching T, Zhu X and Garmire LX (2018) Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data. PLOS Comput Biol 14:e1006076.
Chollet F (2015) Keras. https://github.com/fchollet/keras.
Cirillo D, Ponce-de-Leon M and Valencia A (2021) Algorithmic complexity in computational biology: basics, challenges and limitations. doi: 10.48550/arXiv.1811.07312
Contreras-Moreira B, Filippi CV, Naamati G, García Girón C, Allen JE and Flicek P (2021) K-mer counting and curated libraries drive efficient annotation of repeats in plant genomes. Plant Genome 14:e20143.
Cui Y, Jia M, Lin T-Y, Song Y and Belongie S (2019) Class-Balanced Loss Based on Effective Number of Samples. doi: 10.48550/arXiv.1901.05555
da Cruz MHP, Domingues DS, Saito PTM, Paschoal AR and Bugatti PH (2021) TERL: classification of transposable elements by convolutional neural networks. Brief Bioinform 22:bbaa185.
Deneweth J, Van de Peer Y and Vermeirssen V (2022) Nearby transposable elements impact plant stress gene regulatory networks: a meta-analysis in A. thaliana and S. lycopersicum. BMC Genomics 23:18.
Di Stefano L (2022) All Quiet on the TE Front? The Role of Chromatin in Transposable Element Silencing. Cells 11:2501.
Drongitis D, Aniello F, Fucci L and Donizetti A (2019) Roles of Transposable Elements in the Different Layers of Gene Expression Regulation. Int J Mol Sci 20:5755.
Flutre T, Duprat E, Feuillet C and Quesneville H (2011) Considering Transposable Element Diversification in De Novo Annotation Approaches. PLOS ONE 6:e16526.
Fu L, Niu B, Zhu Z, Wu S and Li W (2012) CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinformatics 28:3150–3152.
Gavrilov AD, Jordache A, Vasdani M and Deng J (2018) Preventing Model Overfitting and Underfitting in Convolutional Neural Networks. Int J Softw Sci Comput Intell IJSSCI 10:19–28.
Girgis HZ (2015) Red: an intelligent, rapid, accurate tool for detecting repeats de-novo on the genomic scale. BMC Bioinformatics 16:227.
Goerner-Potvin P and Bourque G (2018) Computational tools to unmask transposable elements. Nat Rev Genet 19:688–704.
Goubert C, Craig RJ, Bilat AF, Peona V, Vogan AA and Protasio AV (2022) A beginner's guide to manual curation of transposable elements. Mob DNA 13:7.
Han G, Zhang N, Xu J, Jiang H, Ji C, Zhang Z, Song Q, Stanley D, Fang J and Wang J (2019) Characterization of a novel Helitron family in insect genomes: insights into classification, evolution and horizontal transfer. Mob DNA 10:25.
Han W, Zhang Z, Zhang Y, Yu J, Chiu C-C, Qin J, Gulati A, Pang R and Wu Y (2020) ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context. doi: 10.48550/arXiv.2005.03191

Hayward A and Gilbert C (2022) Transposable elements. Curr Biol 32:R904–R909.
Hernandez J, Carrasco-Ochoa JA and Martínez-Trinidad JF (2013) An Empirical Study of Oversampling and Undersampling for Instance Selection Methods on Imbalance Datasets. In: Ruiz-Shulcloper J and Sanniti di Baja G (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer, Berlin, Heidelberg, pp 262–269
Hrinchuk O, Khrulkov V, Mirvakhabova L, Orlova E and Oseledets I (2020) Tensorized Embedding Layers for Efficient Model Compression. doi: 10.48550/arXiv.1901.10787
Hu K, Xu K, Wen J, Yi B, Shen J, Ma C, Fu T, Ouyang Y and Tu J (2019) Helitron distribution in Brassicaceae and whole Genome Helitron density as a character for distinguishing plant species. BMC Bioinformatics 20:354.
Ison J, Ménager H, Brancotte B, Jaaniso E, Salumets A, Raček T, Lamprecht A-L, Palmblad M, Kalaš M, Chmura P et al. (2020) Community curation of bioinformatics software and data resources. Brief Bioinform 21:1697–1705.
Jedlicka P, Lexa M and Kejnovsky E (2020) What Can Long Terminal Repeats Tell Us About the Age of LTR Retrotransposons, Gene Conversion and Ectopic Recombination? Front. Plant Sci. 11:
Ji Y, Zhou Z, Liu H and Davuluri RV (2021) DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Bioinformatics 37:2112–2120.
Johnson JM and Khoshgoftaar TM (2019) Deep Learning and Data Sampling with Imbalanced Big Data. 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI). pp 175–183
Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Žídek A, Potapenko A et al. (2021) Highly accurate protein structure prediction with AlphaFold. Nature 596:583–589.
Kapitonov VV, Tempel S and Jurka J (2009) Simple and fast classification of non-LTR retrotransposons based on phylogeny of their RT domain protein sequences. Gene 448:207–213.
Krampis K (2022) Democratizing bioinformatics through easily accessible software platforms for non-experts in the field. BioTechniques 72:36–38.
Krawczyk B, Woźniak M and Herrera F (2014) Weighted one-class classification for different types of minority class examples in imbalanced data. 2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). pp 337–344
Lawlor B and Sleator RD (2020) The democratization of bioinformatics: A software engineering perspective. GigaScience 9:giaa063.
Li W and Godzik A (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics 22:1658–1659.
Li Y, Huang C, Ding L, Li Z, Pan Y and Gao X (2019) Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. Methods 166:4–21.
Li Y and Yang T (2018) Word Embedding for Understanding Natural Language: A Survey. In: Srinivasan S (ed) Guide to Big Data Applications. Springer International Publishing, Cham, pp 83–104
LIANG Jie CJ (2019) One-hot encoding and convolutional neural network based anomaly detection. J Tsinghua Univ Technol 59:523–529.
Liu B, Blekas K and Tsoumakas G (2022) Multi-label sampling based on local label imbalance. Pattern Recognit 122:108294.
Lopik J van, Trapotsi M-A, Hannon GJ, Bornelöv S and Nicholson BC (2023) Unistrand piRNA clusters are an evolutionary conserved mechanism to suppress

endogenous retroviruses across the Drosophila genus. 2023.02.27.530199.
Luo H, Xiong C, Fang W, Love PED, Zhang B and Ouyang X (2018) Convolutional neural networks: Computer vision-based workforce activity assessment in construction. Autom Constr 94:282–289.
Martorell-Marugán J, Tabik S, Benhammou Y, Val C del, Zwir I, Herrera F and Carmona-Sáez P (2019) Deep Learning in Omics Data Analysis and Precision Medicine. Exon Publ 37–53.
Meena D, Behera BK, Das P, Prusty AK, Kumar S and MEENA M (2012) Transposable elements: Strategies and mechanism of transposition in Danio rerio, a genetic model. J Bio-Sci 7:223–229.
Mérel V, Boulesteix M, Fablet M and Vieira C (2020) Transposable elements in Drosophila. Mob DNA 11:23.
Mikolov T, Chen K, Corrado G and Dean J (2013a) Efficient Estimation of Word Representations in Vector Space. doi: 10.48550/arXiv.1301.3781
Mikolov T, Sutskever I, Chen K, Corrado G s and Dean J (2013b) Distributed Representations of Words and Phrases and their Compositionality. Adv. Neural Inf. Process. Syst. 26:
Mombach DM, da Fontoura Gomes TMF and Loreto ELS (2022) Stress does not induce a general transcription of transposable elements in Drosophila. Mol Biol Rep 49:9033–9040.
Ono H, Ishii K, Kozaki T, Ogiwara I, Kanekatsu M and Yamada T (2015) Removal of redundant contigs from de novo RNA-Seq assemblies via homology search improves accurate detection of differentially expressed genes. BMC Genomics 16:1031.
Ou S, Su W, Liao Y, Chougule K, Agda JRA, Hellinga AJ, Lugo CSB, Elliott TA, Ware D, Peterson T et al. (2019) Benchmarking transposable element annotation methods for creation of a streamlined, comprehensive pipeline. Genome Biol 20:275.
Oubounyt M, Louadi Z, Tayara H and Chong KT (2019) DeePromoter: Robust Promoter Predictor Using Deep Learning. Front. Genet. 10:
Pearson WR (2013) An Introduction to Sequence Similarity ("Homology") Searching. Curr Protoc Bioinforma Ed Board Andreas Baxevanis Al 0 3:10.1002/0471250953.bi0301s42.
Peng L, Peng M, Liao B, Huang G, Li W and Xie D (2018) The Advances and Challenges of Deep Learning Application in Biological Big Data Processing. Curr Bioinforma 13:352–359.
Pennington J, Socher R and Manning C (2014) GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar, pp 1532–1543
Permal E, Flutre T and Quesneville H (2012) Roadmap for annotating transposable elements in eukaryote genomes. Methods Mol Biol Clifton NJ 859:53–68.
Petersen M, Armisén D, Gibbs RA, Hering L, Khila A, Mayer G, Richards S, Niehuis O and Misof B (2019) Diversity and evolution of the transposable element repertoire in arthropods with particular reference to insects. BMC Ecol Evol 19:11.
Rawat W and Wang Z (2017) Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Comput 29:2352–2449.
Riehl K, Riccio C, Miska EA and Hemberg M (2022) TransposonUltimate: software for transposon classification, annotation and detection. Nucleic Acids Res 50:e64.
Rodriguez F and Arkhipova IR (2023) An overview of best practices for transposable element identification, classification and annotation in eukaryotic genomes. Methods

Mol Biol Clifton NJ 2607:1–23.
Rodriguez M and Makałowski W (2022) Software evaluation for de novo detection of transposons. Mob DNA 13:14.
Rodríguez P, Bautista MA, Gonzàlez J and Escalera S (2018) Beyond one-hot encoding: Lower dimensional target embedding. Image Vis Comput 75:21–31.
Rossi M, Araujo PG and Van Sluys M-A (2001) Survey of transposable elements in sugarcane expressed sequence tags (ESTs). Genet Mol Biol 24:147–154.
Rubin PM, Loreto ELS, Carareto CMA and Valente VLS (2011) The copia retrotransposon and horizontal transfer in Drosophila willistoni. Genet Res 93:175–180.
Saini M and Susan S (2022) Diabetic retinopathy screening using deep learning for multi-class imbalanced datasets. Comput Biol Med 149:105989.
Searls DB (1992) The Linguistics of DNA. Am Sci 80:579–591.
Searls DB (2002) The language of genes. Nature 420:211–217.
Serrato-Capuchina A and Matute DR (2018) The Role of Transposable Elements in Speciation. Genes 9:254.
Sexton CE and Han MV (2019) Paired-end mappability of transposable elements in the human genome. Mob DNA 10:29.
Shahinfar S, Meek P and Falzon G (2020) “How many images do I need?” Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. Ecol Inform 57:101085.
Skipper K, Andersen P, Sharma N and Mikkelsen J (2013) DNA transposon-based gene vehicles - Scenes from an evolutionary drive. J Biomed Sci 20:92.
Sotero-Caio CG, Platt RN II, Suh A and Ray DA (2017) Evolution and Diversity of Transposable Elements in Vertebrate Genomes. Genome Biol Evol 9:161–177.
Stitzer MC, Anderson SN, Springer NM and Ross-Ibarra J (2021) The genomic ecosystem of transposable elements in maize. PLoS Genet 17:e1009768.
Storer JM, Hubley R, Rosen J and Smit AFA (2022) Methodologies for the De novo Discovery of Transposable Element Families. Genes 13:709.
Thompson PJ, Macfarlan TS and Lorincz MC (2016) Long terminal repeats: from parasitic elements to building blocks of the transcriptional regulatory repertoire. Mol Cell 62:766–776.
Umarov R, Kuwahara H, Li Y, Gao X and Solovyev V (2019) Promoter analysis and prediction in the human genome using sequence-based deep learning models. Bioinformatics 35:2730–2737.
Vitte C and Panaud O (2003) Formation of Solo-LTRs Through Unequal Homologous Recombination Counterbalances Amplifications of LTR Retrotransposons in Rice Oryza sativa L. Mol Biol Evol 20:528–540.
Wahab A, Tayara H, Xuan Z and Chong KT (2021) DNA sequences performs as natural language processing by exploiting deep learning algorithm for the identification of N4-methylcytosine. Sci Rep 11:212.
Walker MWG, Klompe SE, Zhang DJ and Sternberg SH (2023) Transposon mutagenesis libraries reveal novel molecular requirements during CRISPR RNA-guided DNA integration. 2023.01.19.524723.
Wang H, Cui Z, Chen Y, Avidan M, Abdallah AB and Kronzer A (2018) Predicting Hospital Readmission via Cost-Sensitive Deep Learning. IEEE/ACM Trans Comput Biol Bioinform 15:1968–1978.
Wang W and Gang J (2018) Application of Convolutional Neural Network in Natural Language Processing. 2018 International Conference on Information Systems and

Computer Aided Education (ICISCAE). pp 64–70
Wells JN and Feschotte C (2020) A Field Guide to Eukaryotic Transposable Elements. Annu Rev Genet 54:539–561.
Wicker T, Sabot F, Hua-Van A, Bennetzen JL, Capy P, Chalhoub B, Flavell A, Leroy P, Morgante M, Panaud O et al. (2007) A unified classification system for eukaryotic transposable elements. Nat Rev Genet 8:973–982.
Xiong W, He L, Lai J, Dooner HK and Du C (2014) HelitronScanner uncovers a large overlooked cache of Helitron transposons in many plant genomes. Proc Natl Acad Sci 111:10263–10268.
Xu C, Zhao P, Liu Y, Xu J, S.Sheng VSS, Cui Z, Zhou X and Xiong H (2019) Recurrent Convolutional Neural Network for Sequential Recommendation. The World Wide Web Conference. Association for Computing Machinery, New York, NY, USA, pp 3398–3404
Yan H, Bombarely A and Li S (2020) DeepTE: a computational method for de novo classification of transposons with convolutional neural network. Bioinformatics 36:4269–4275.
You FM, Cloutier S, Shan Y and Ragupathy R (2015) LTR Annotator: Automated Identification and Annotation of LTR Retrotransposons in Plant Genomes. Int J Biosci Biochem Bioinforma 5:165–174.
Zhang L, Yan L, Jiang J, Wang Y, Jiang Y, Yan T and Cao Y (2014) The structure and retrotransposition mechanism of LTR-retrotransposons in the asexual yeast Candida albicans. Virulence 5:655–664.
Zhang Q, Yang LT, Chen Z and Li P (2018) A survey on deep learning for big data. Inf Fusion 42:146–157.
Zielezinski A, Vinga S, Almeida J and Karlowski WM (2017) Alignment-free sequence comparison: benefits, applications, and tools. Genome Biol 18:186.
Zou J, Huss M, Abid A, Mohammadi P, Torkamani A and Telenti A (2019) A primer on deep learning in genomics. Nat Genet 51:12–18.
Zrimec J, Börlin CS, Buric F, Muhammad AS, Chen R, Siewers V, Verendel V, Nielsen J, Töpel M and Zelezniak A (2020) Deep learning suggests that gene expression is encoded in all parts of a co-evolving interacting gene regulatory structure. Nat Commun 11:6141.

Table 1. A summary of the datasets used to train HamleTE workflow's classifier models.

Dataset	Total sequences	Number of classes	Training/validation split	Class imbalance
TE/non-TE	18914	3	80/20	No
TE class	19064	2	80/20	No
LTR/non-LTR	29989	2	80/20	Yes
Class II superfamilies	14325	5	90/10	No
LTR superfamilies	11358	3	90/10	No
non-LTR superfamilies	30780	7	90/10	Yes

Supplementary table 1. Hyperparameter layer values for each model. Layers or hyperparameters not shown in the table were using default values. The semicolons are separating the value for each layer. The model architecture is represented in Figure 1.

Model	Embedding		Conv1D			MaxPooling1D	Dense		Dropout
	Embedding size	Vocabulary size	Filters	Kernel size	Activation function	Pool size	Neurons	Activation function	Rate
TE/non-TE	12	6	128;64;64	7	ReLU	7;7;7	32;4	ReLU;Softmax	0.2
TE Class	12	6	64;32;32	7	ReLU	7;7;7	24;3	ReLU;Softmax	0.2
LTR/non-LTR	12	6	32;24;24	7	ReLU	7;7;7	32;3	ReLU;Softmax	0.2
Class II superfamilies	12	6	32;64;64	12	ReLU	7;7;7	24;6	ReLU;Softmax	0.4
LTR superfamilies	12	6	128;32;32	12	ReLU	7;7;7	32;4	ReLU;Softmax	0.4
Non-LTR superfamilies	12	6	32;128;128	7	ReLU	7;7;7	128;8	ReLU;Softmax	0.4

FIGURES

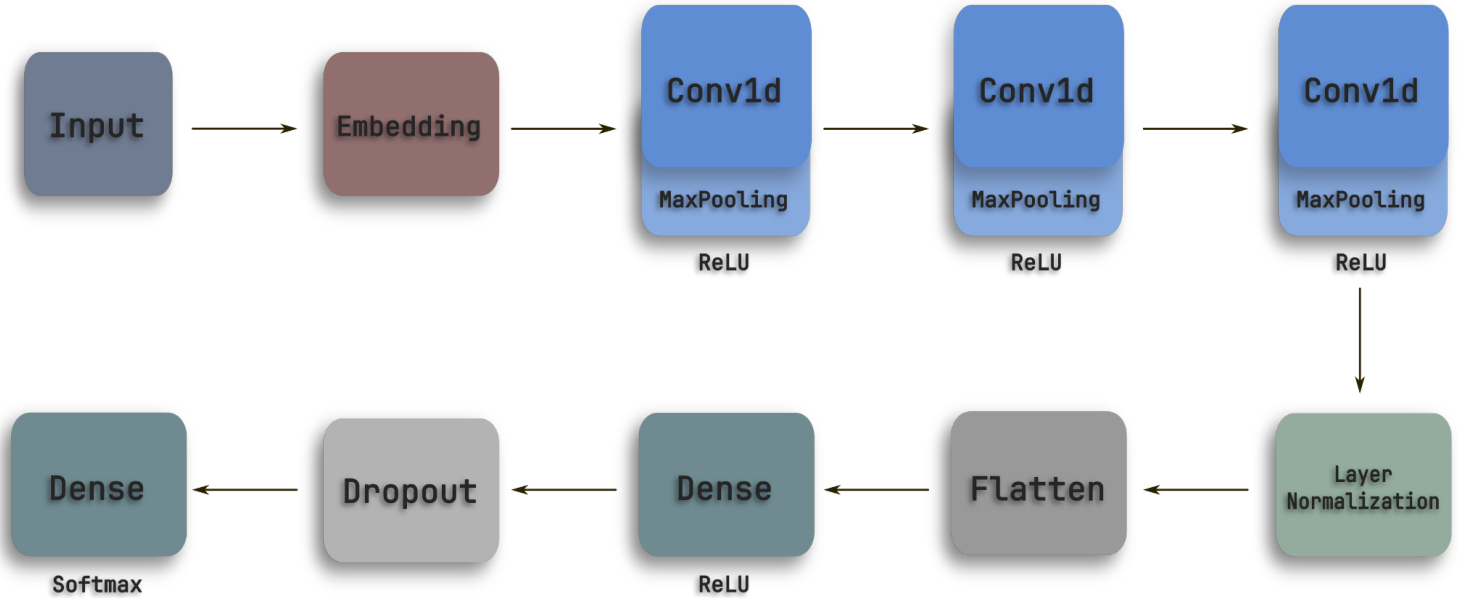


Figure 1. Schematic diagram showing the basic convolutional neural network layers used for all six models.

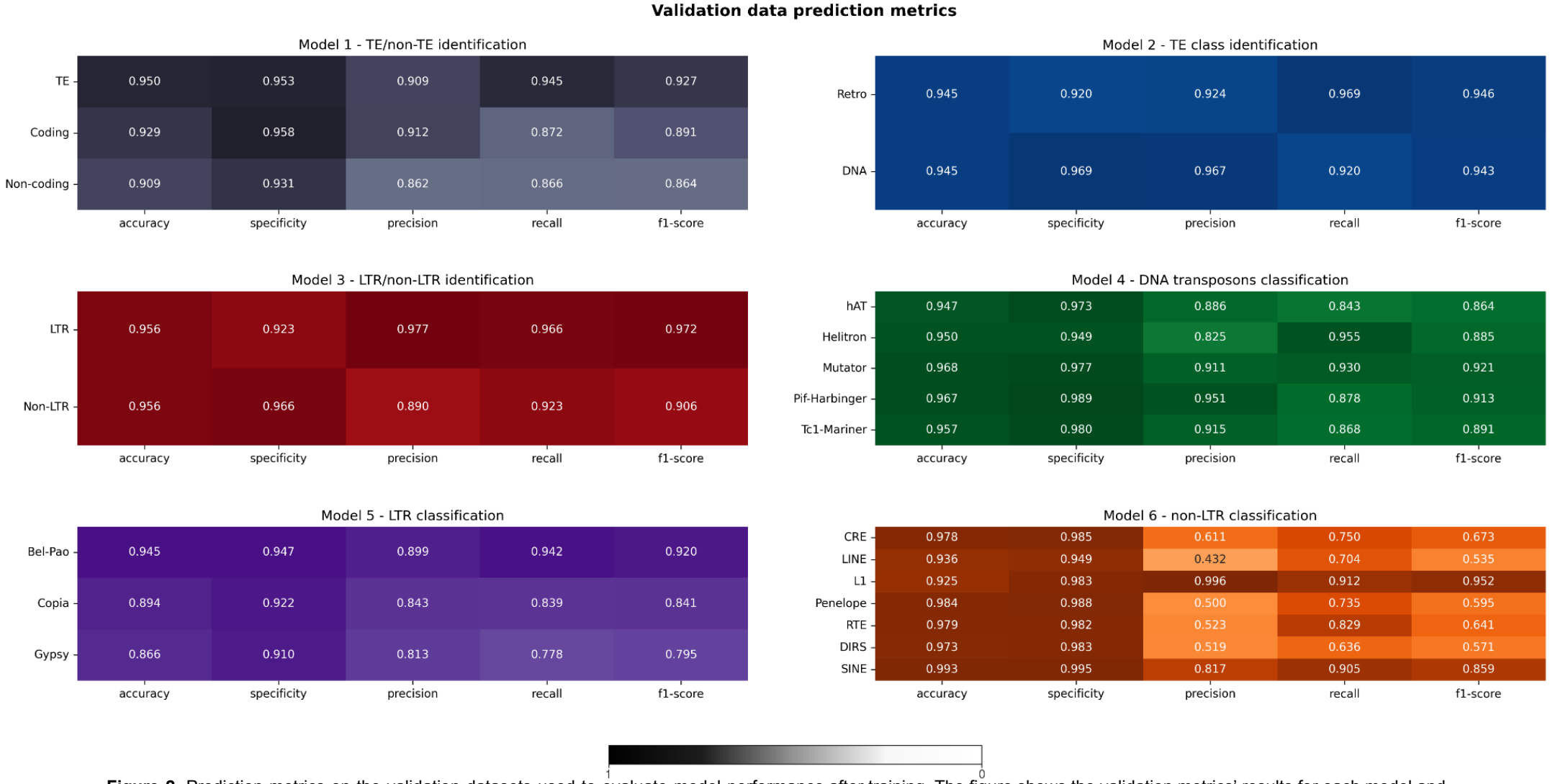
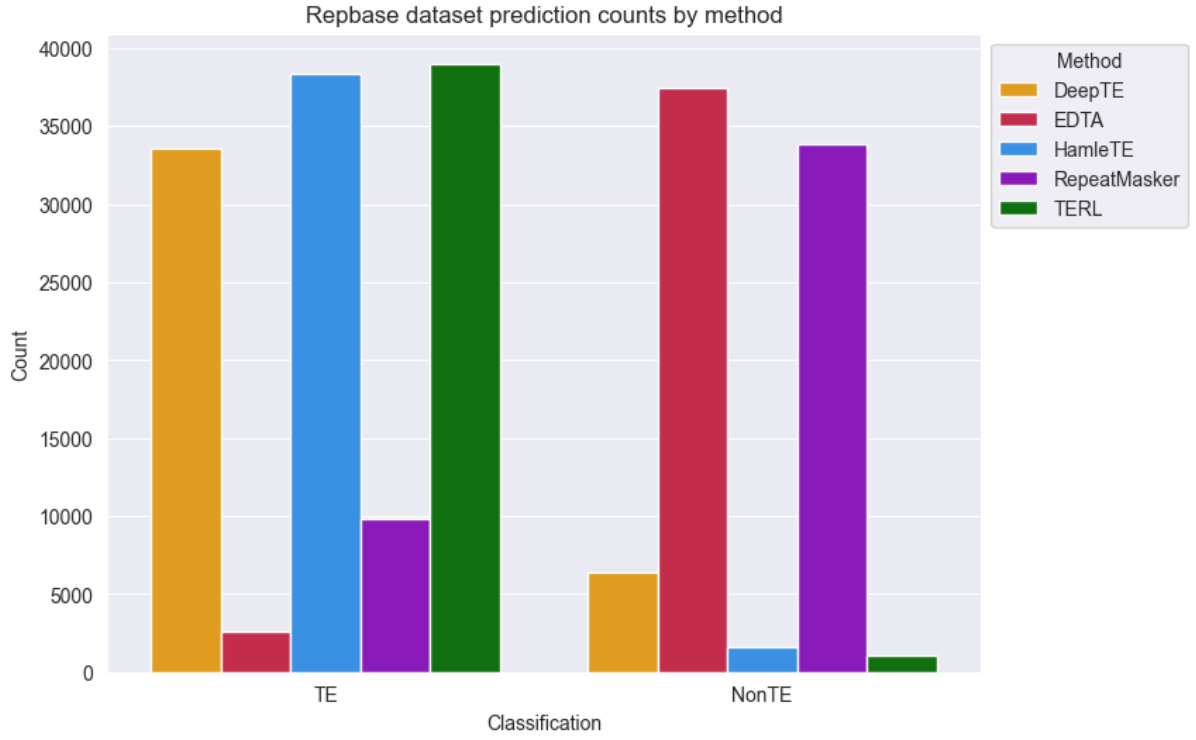


Figure 2 .Prediction metrics on the validation datasets used to evaluate model performance after training. The figure shows the validation metrics' results for each model and TEs category of the corresponding model.

(a)



(b)

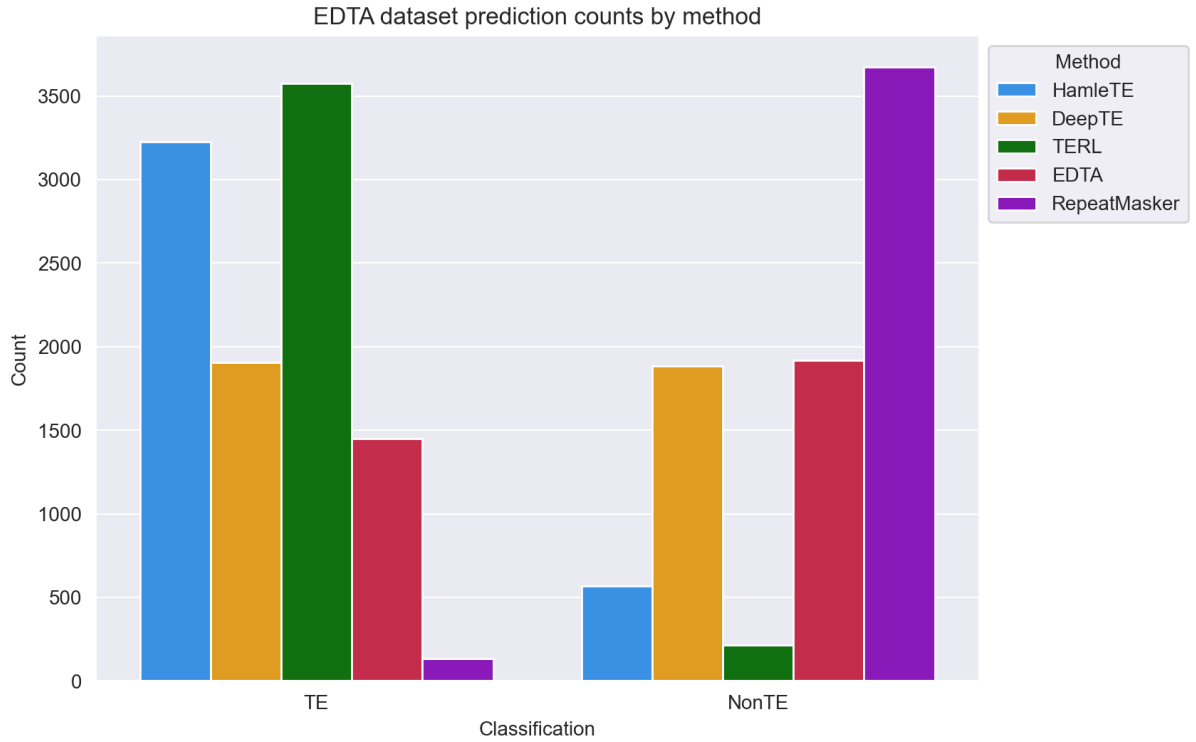


Figure 3. Total number of identified TE by method for the repbase and EDTA datasets. Figure 4a shows the total number of TEs for the Repbase dataset, figure 4b for the EDTA dataset. HamleTE was only behind TERL in the number of sequences correctly identified as TEs.

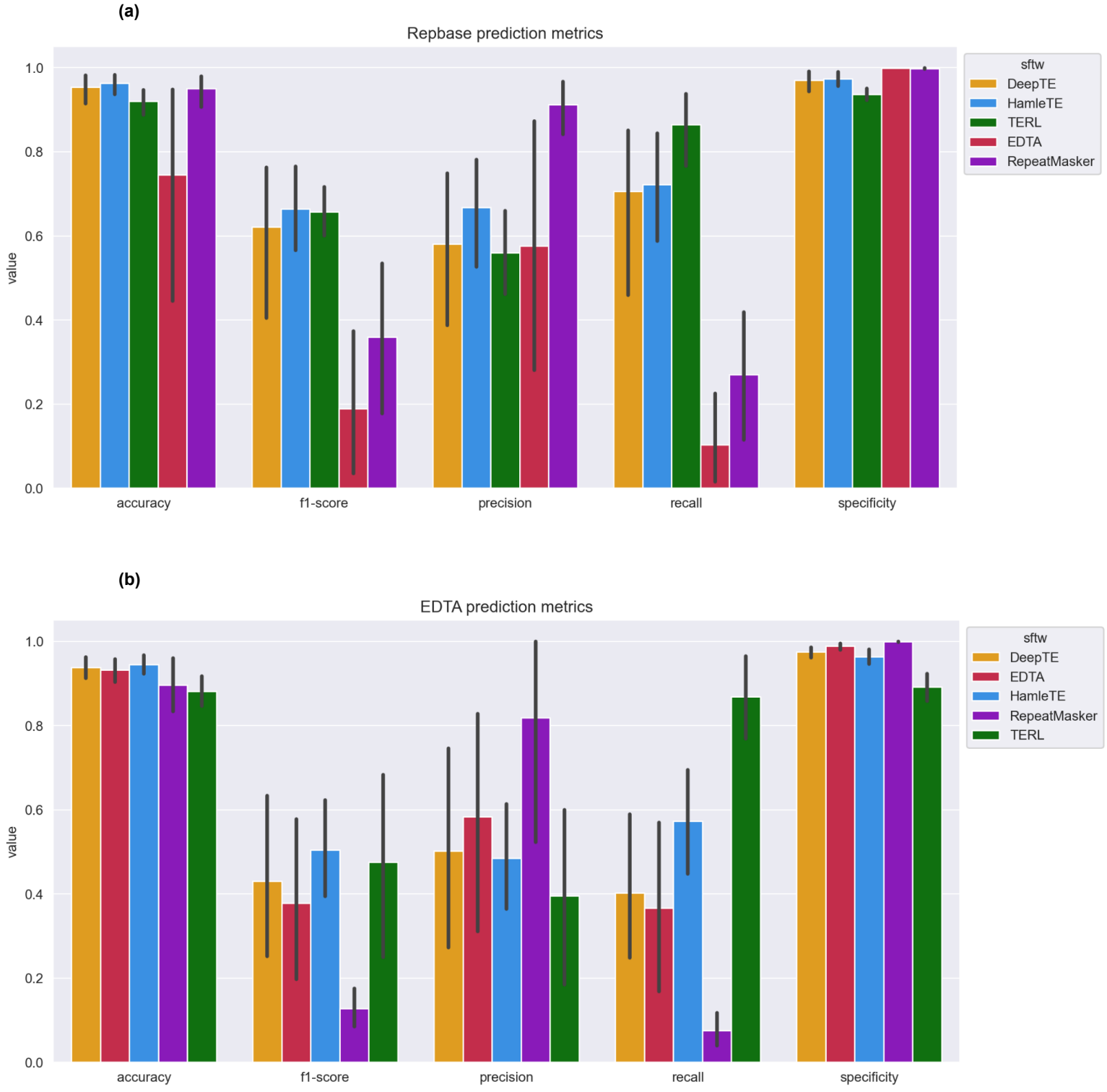
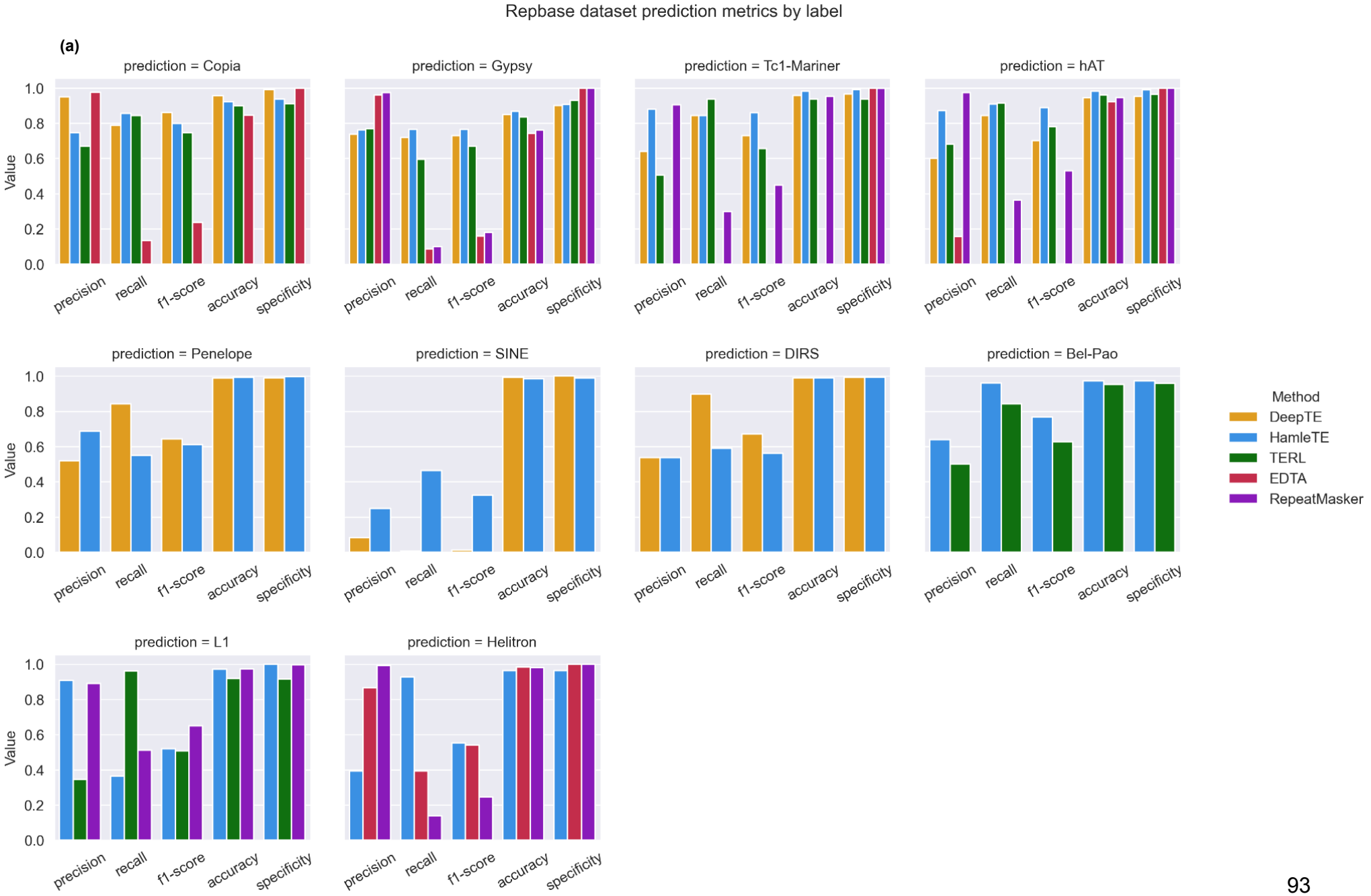


Figure 4. Prediction metrics for the EDTA datasets. Figure 5a shows the metrics for the Repbase dataset and figure 5b for the EDTA dataset. HamleTE had the best f1-score for both datasets, being the most balanced software and, if not the best in other metrics, the close second place.


Repbase dataset prediction metrics by label
(a)
prediction = Copia
prediction = Gypsy
prediction = Tc1-Mariner
prediction = hAT
prediction = Penelope
prediction = SINE
prediction = DIRS
prediction = Bel-Pao
prediction = L1
prediction = Helitron
Value
1.0
0.8
0.6
0.4
0.2
0.0
precision
recall
f1-score
accuracy
specificity
Method
DeepTE
HamleTE
TERL
EDTA
RepeatMasker

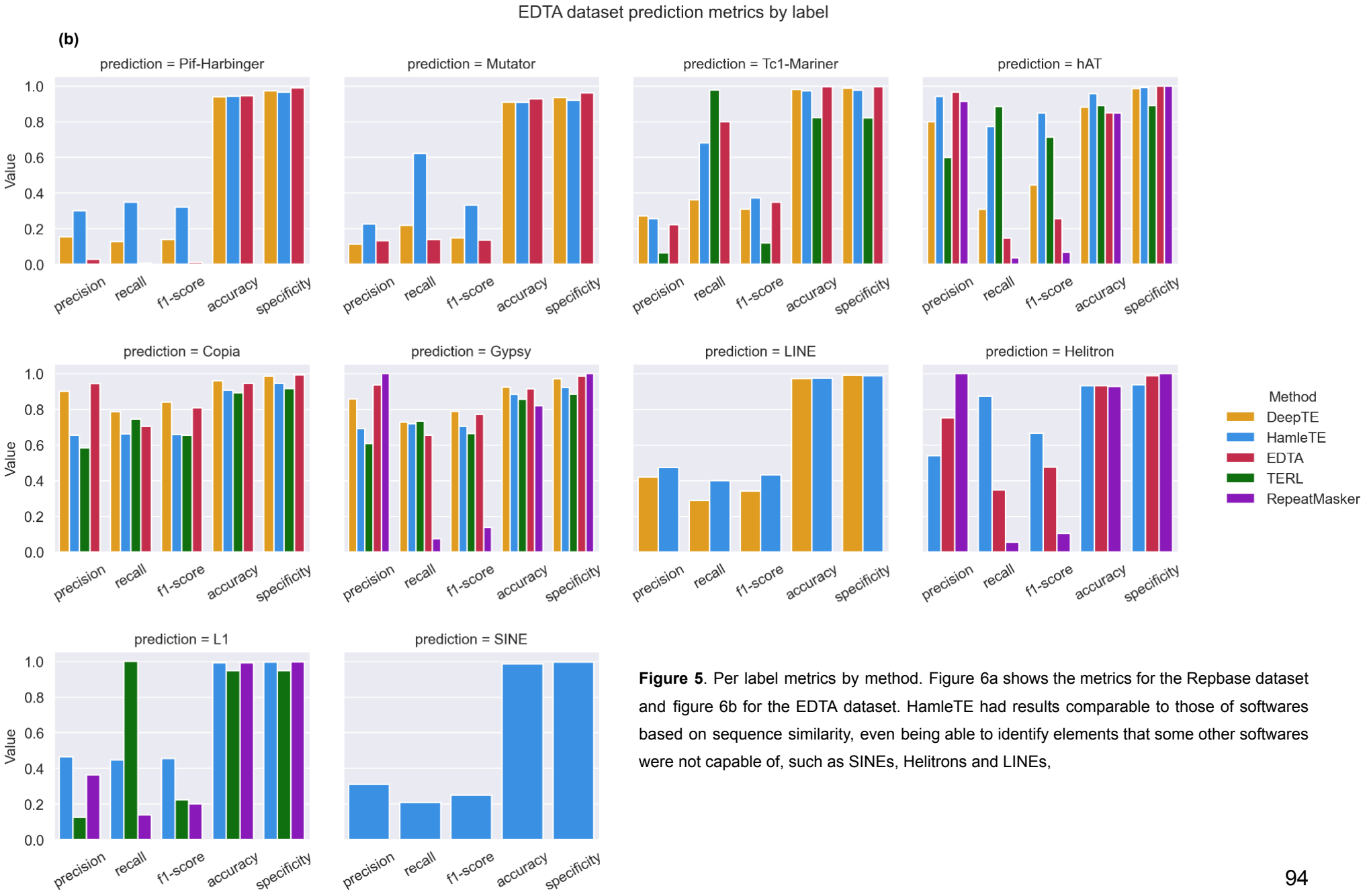


Figure 5. Per label metrics by method. Figure 6a shows the metrics for the Repbase dataset and figure 6b for the EDTA dataset. HamleTE had results comparable to those of softwares based on sequence similarity, even being able to identify elements that some other softwares were not capable of, such as SINEs, Helitrons and LINEs,

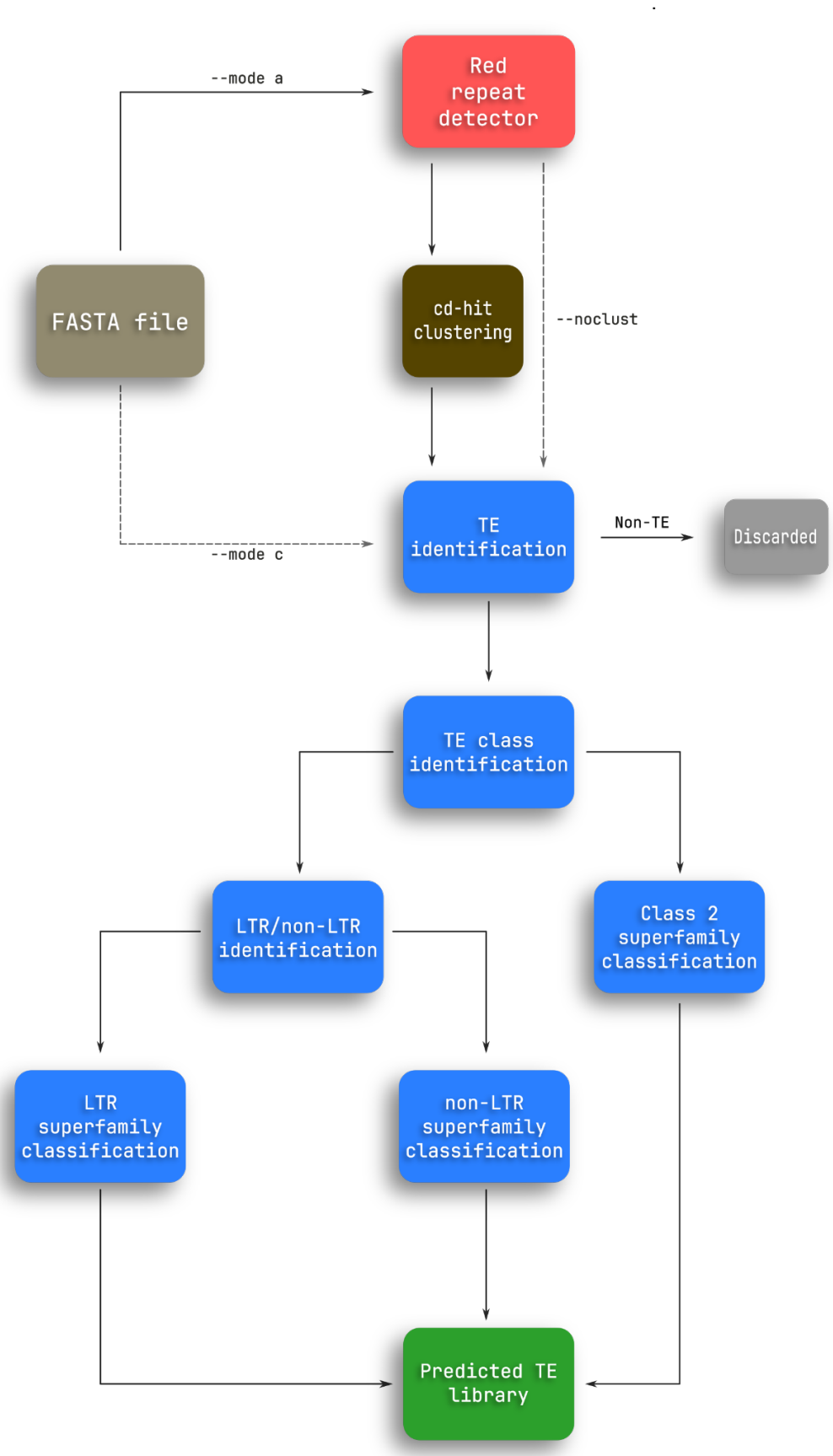


Figure 6. HamleTE's workflow. Dashed arrows are alternatives to the default available for the user. HamleTE uses a hierarchical workflow, which helps to increase accuracy and reduce classification complexity.

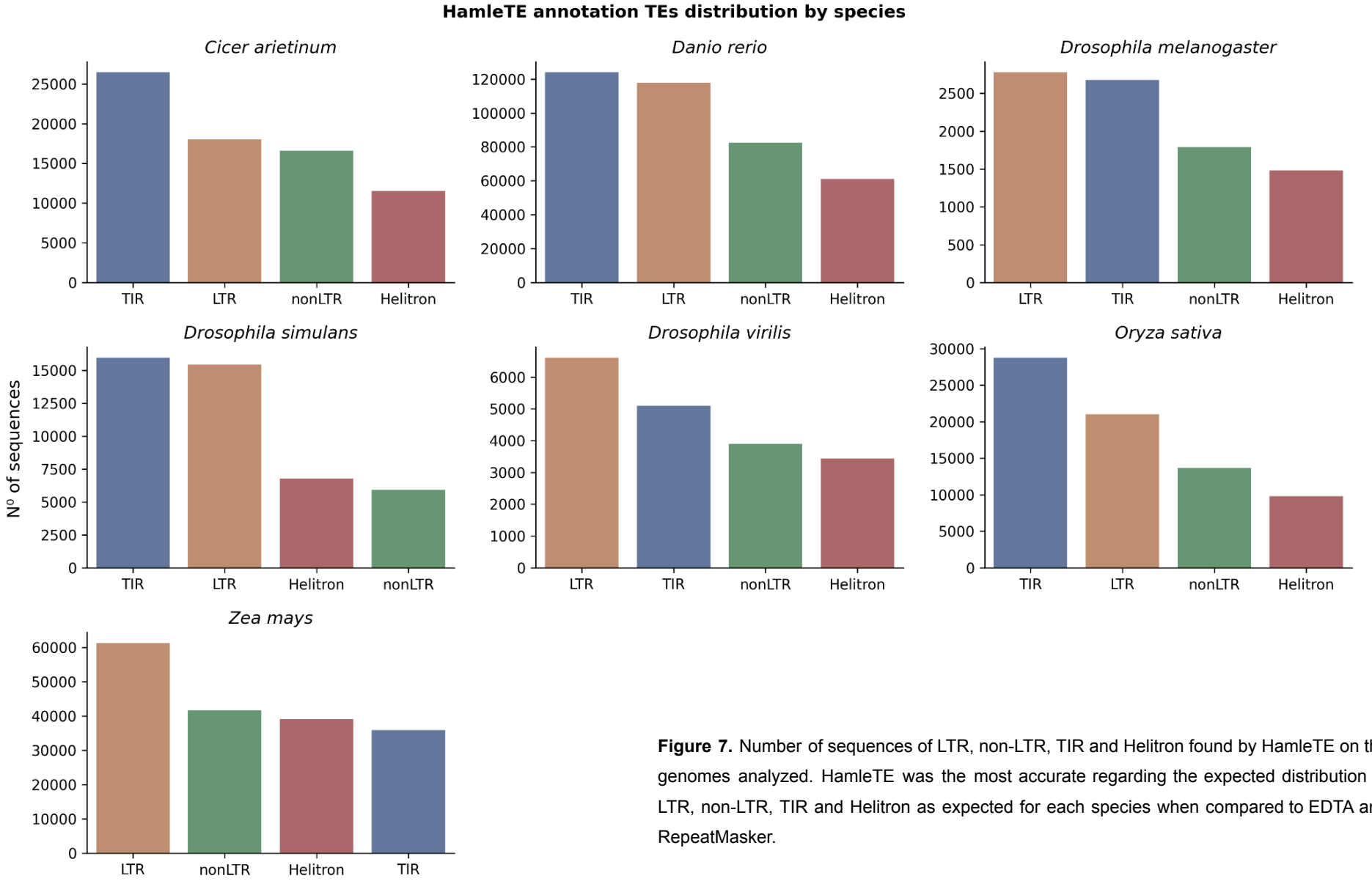


Figure 7. Number of sequences of LTR, non-LTR, TIR and Helitron found by HamleTE on the genomes analyzed. HamleTE was the most accurate regarding the expected distribution of LTR, non-LTR, TIR and Helitron as expected for each species when compared to EDTA and RepeatMasker.

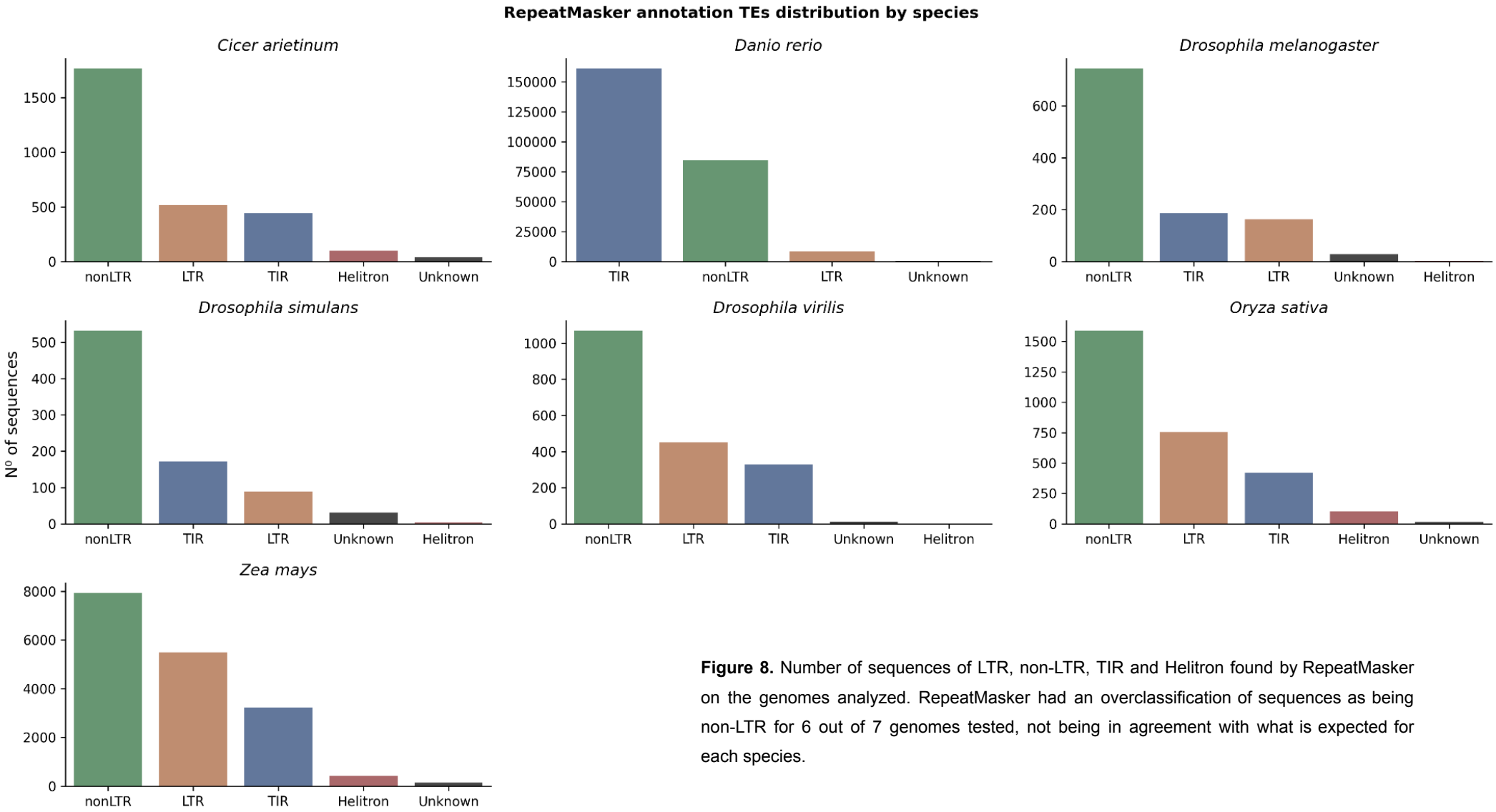


Figure 8. Number of sequences of LTR, non-LTR, TIR and Helitron found by RepeatMasker on the genomes analyzed. RepeatMasker had an overclassification of sequences as being non-LTR for 6 out of 7 genomes tested, not being in agreement with what is expected for each species.

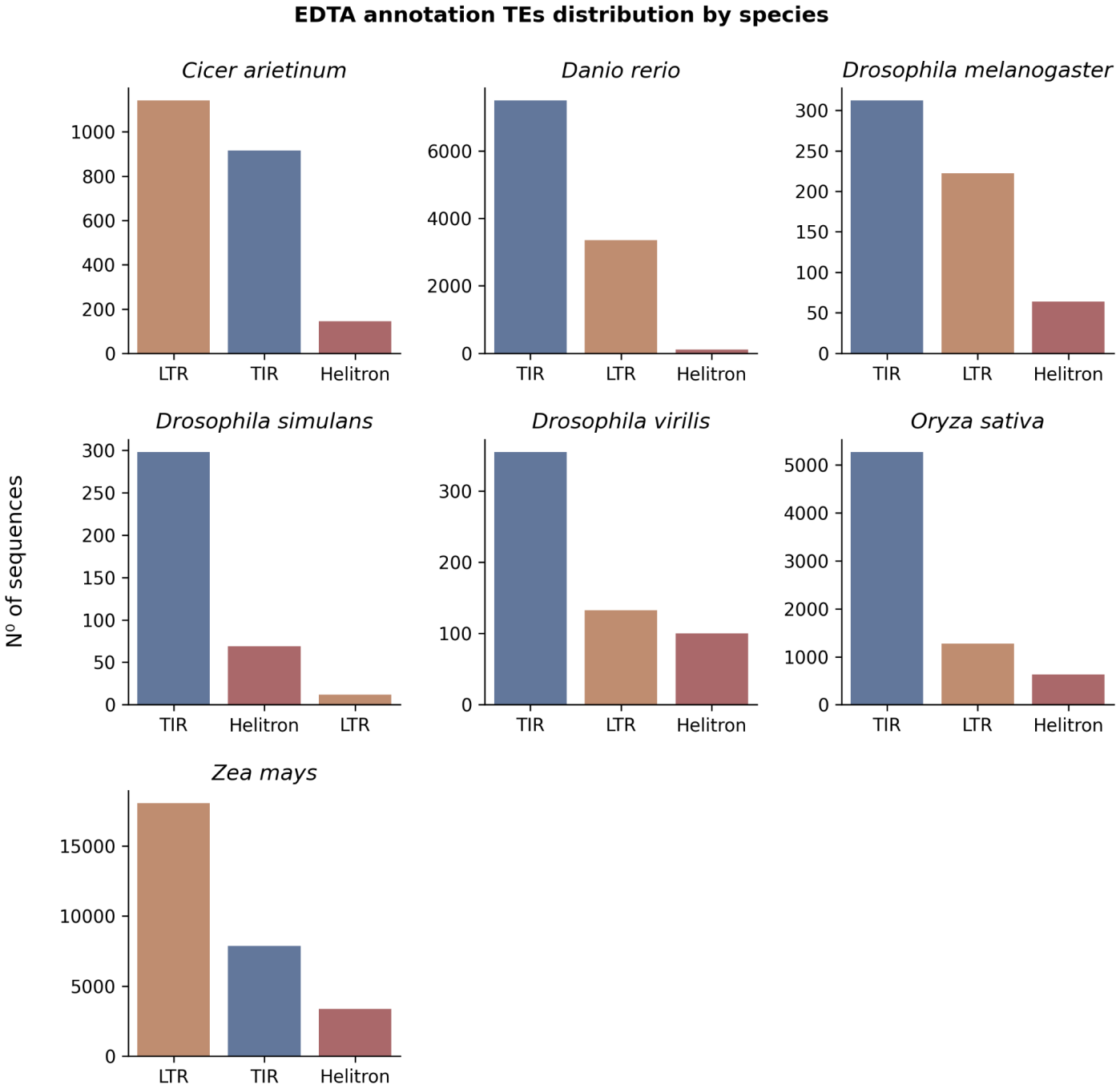


Figure 9. Number of sequences of LTR, non-LTR, TIR and Helitron found by EDTA on the genomes analyzed. EDTA could not identify any non-LTR sequences on the genomes analyzed, and had on overestimation of TIR sequences in its library even for genomes well studied such as *D. melanogaster*.

SUPPLEMENTARY FIGURES

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

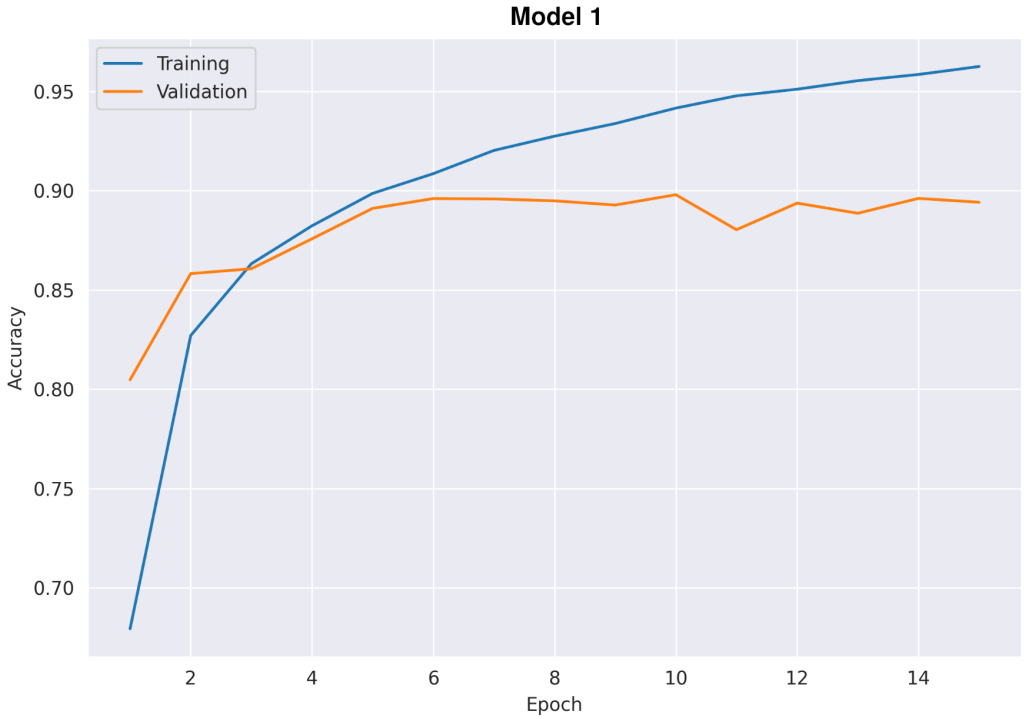
$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

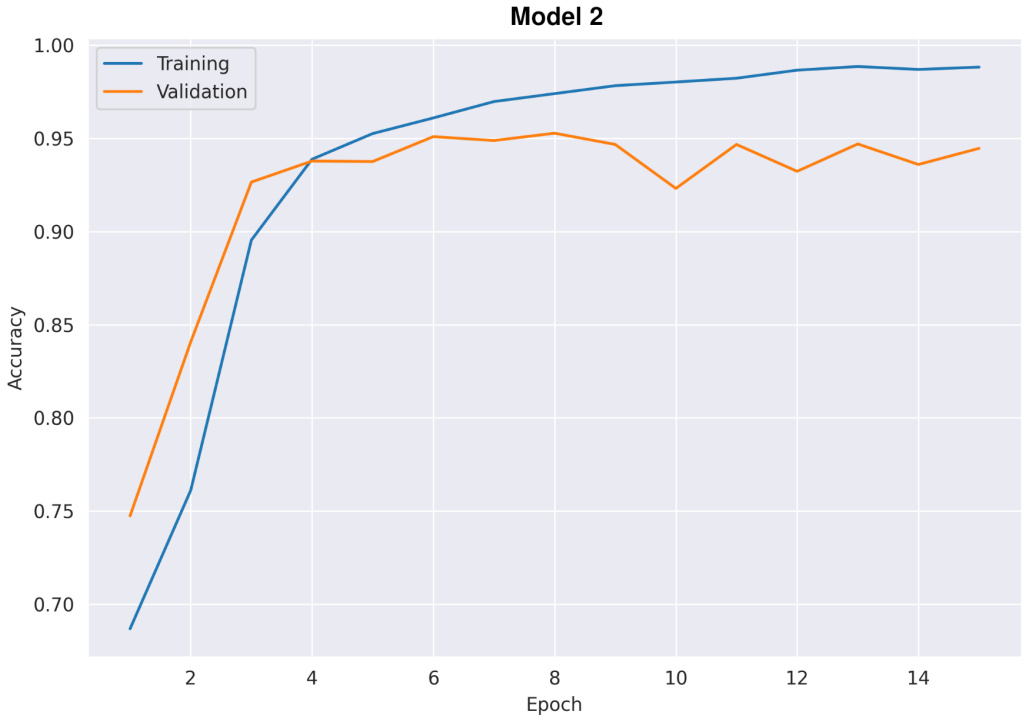
$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

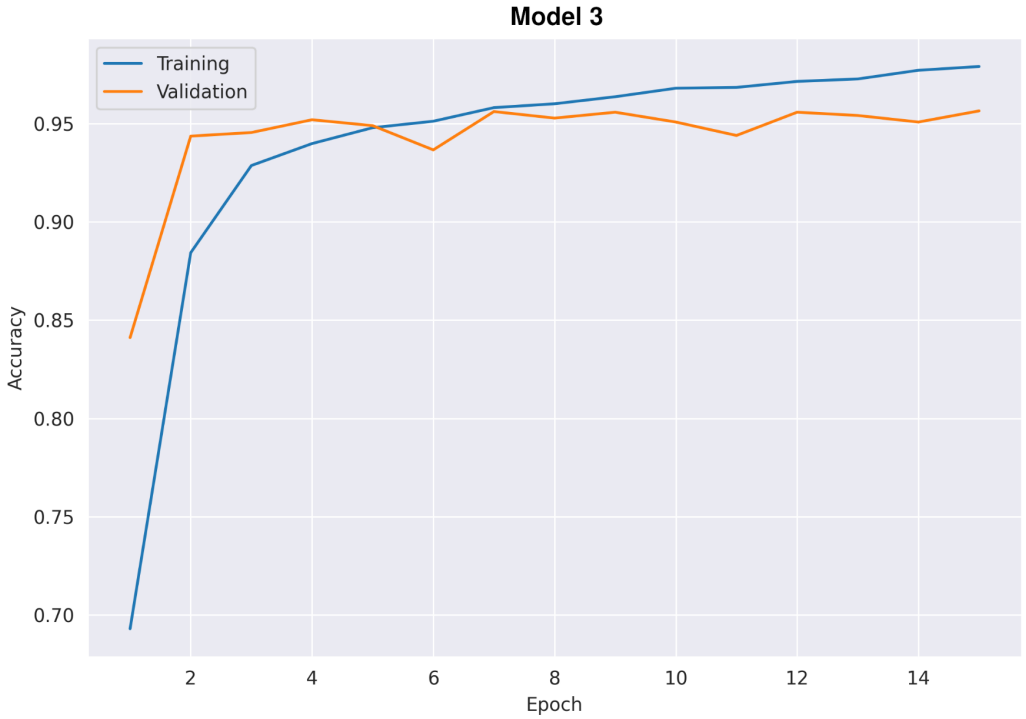
Supplementary figure 1. Benchmark metrics used to assess HamleTE performance. TP: true positive, TF: true negative, FP: false positive, FN: false negative.



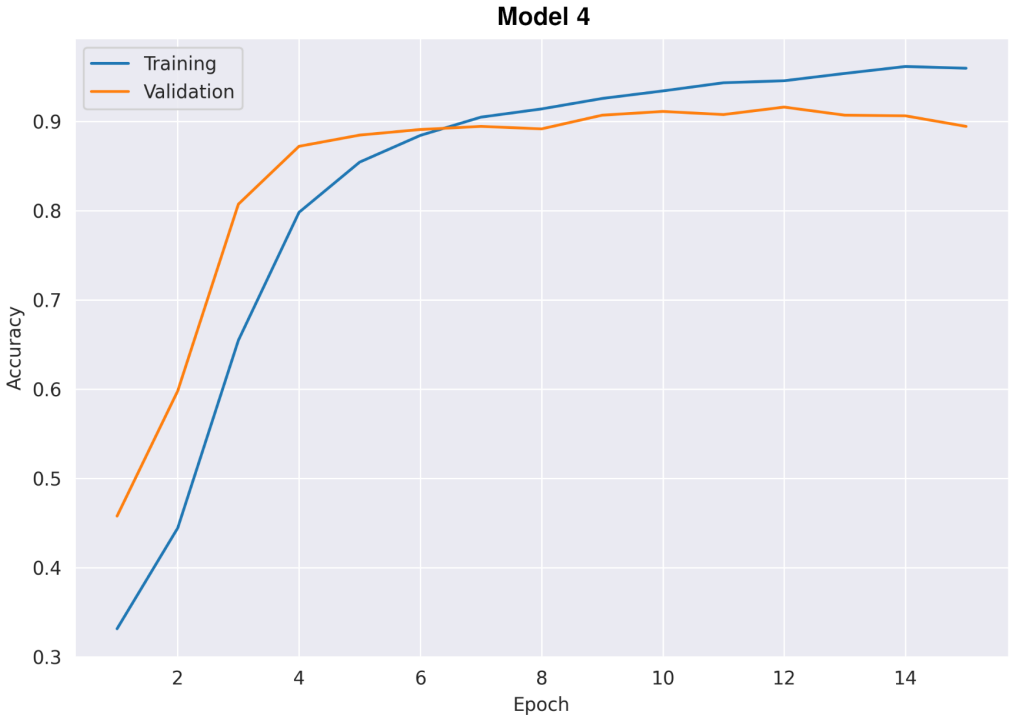
Supplementary Figure 2. Model 1 training and validation accuracy across epochs to learn how to identify TE from non-TE such as protein coding genes and non-coding RNAs. Training accuracy score was 0.96 and validation accuracy had a score of 0.894.



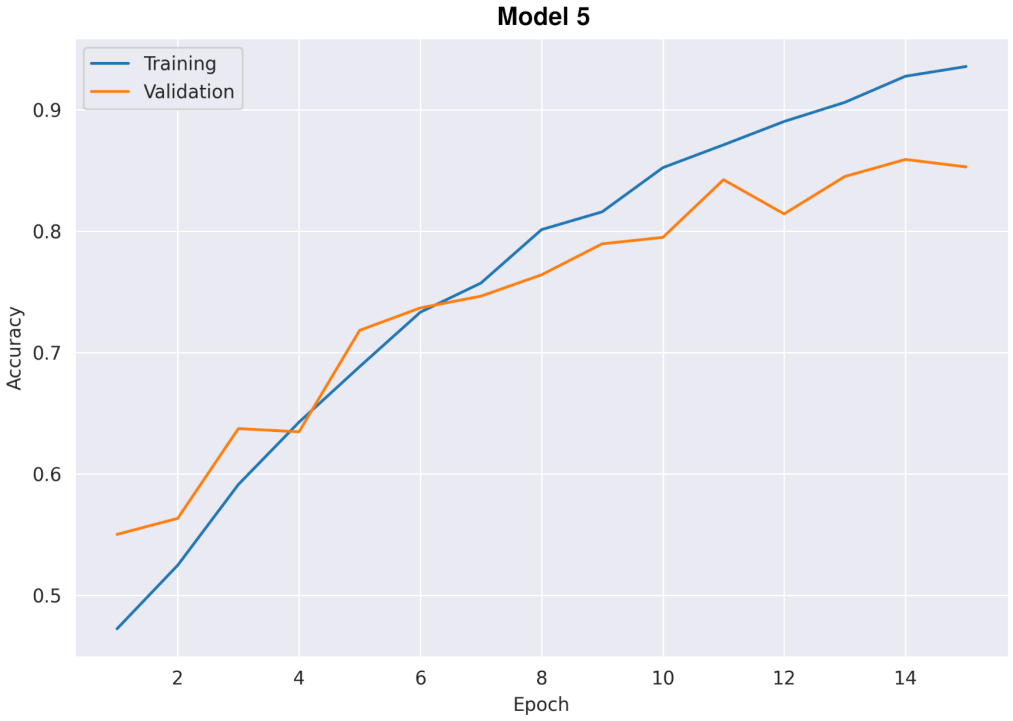
Supplementary Figure 3. Model 2 training and validation accuracy across epochs to learn how to identify class 1 and class 2 TEs. Training accuracy score was 0.988 and validation accuracy had a score of 0.945.



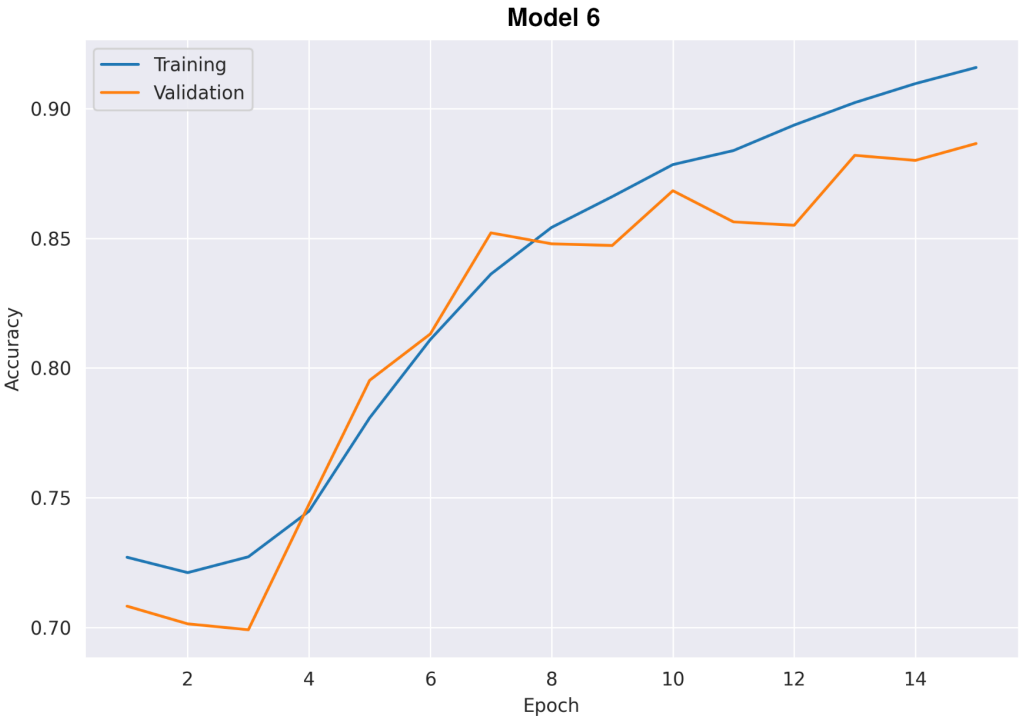
Supplementary Figure 4. Model 3 training and validation accuracy across epochs to learn how to identify LTR and non-LTR elements at superfamily level. Training accuracy score was 0.97 and validation accuracy had a score of 0.95 approximately.



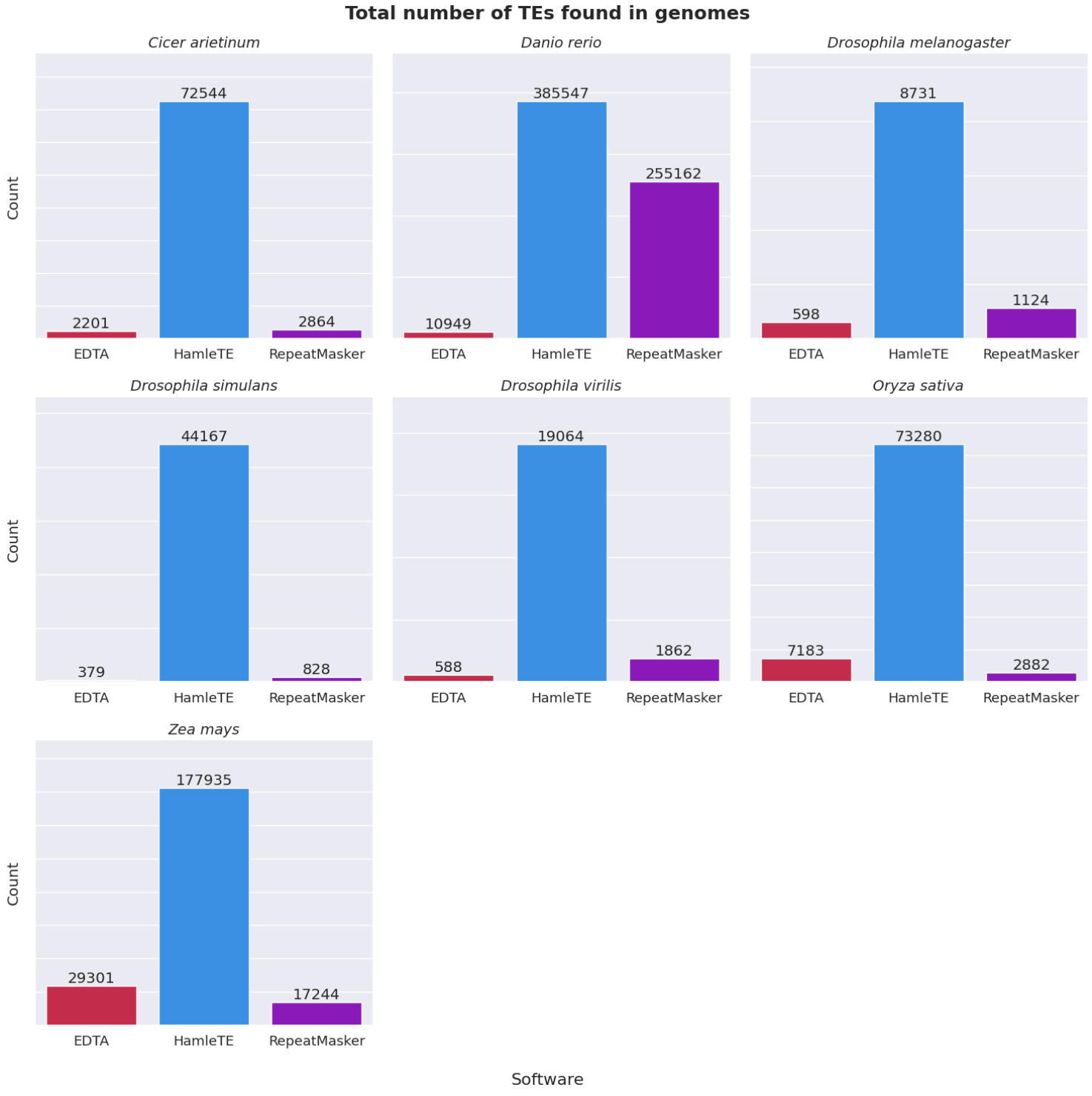
Supplementary Figure 5. Model 4 training and validation accuracy across epochs to learn how to classify class 2 elements (DNA transposons) at superfamily level. Training accuracy score was 0.95 and validation accuracy had a score of 0.894.



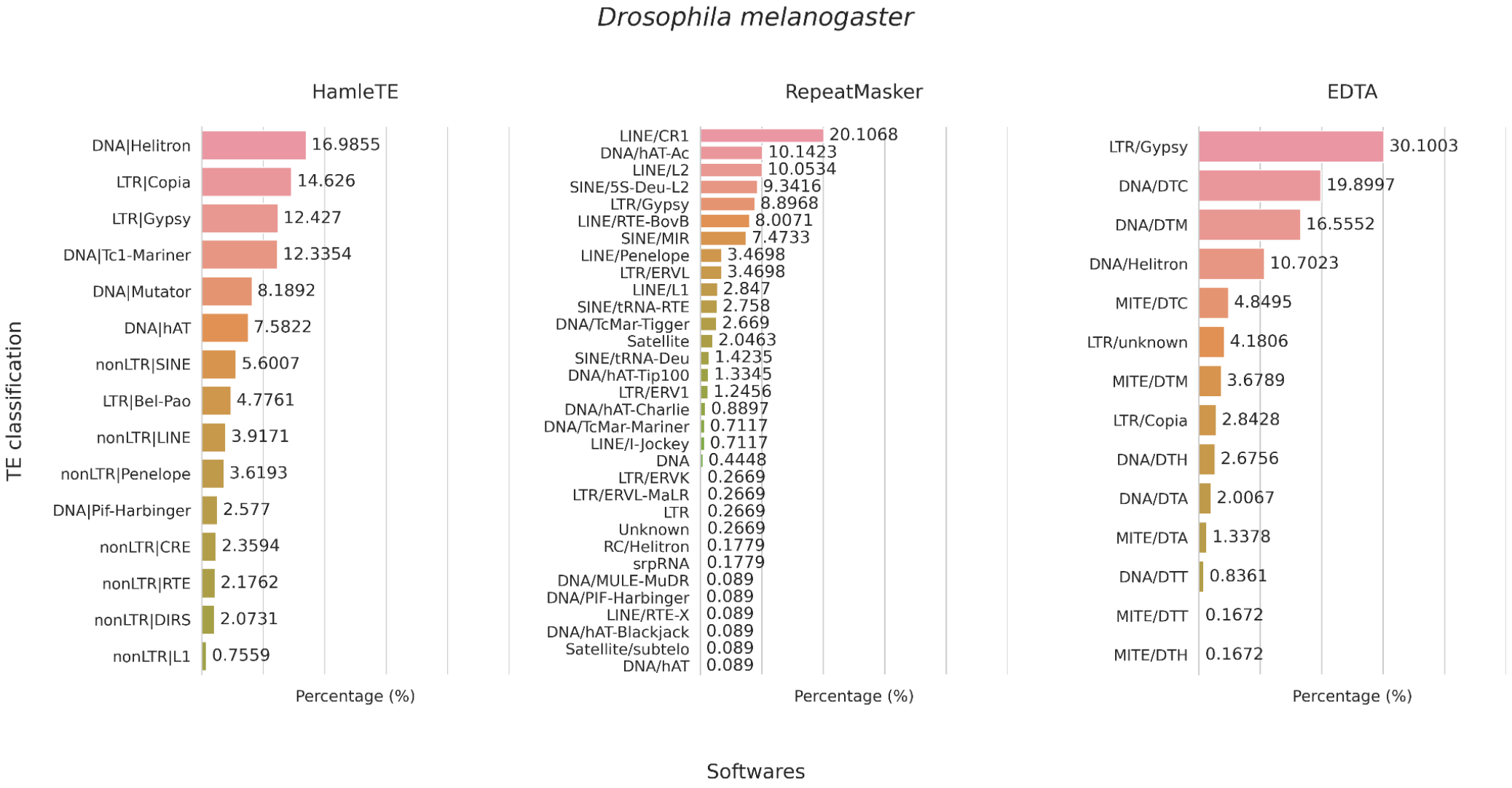
Supplementary Figure 6. Model 5 training and validation accuracy across epochs to learn how to classify LTR elements at superfamily level. Training accuracy score was 0.93 and validation accuracy had a score of 0.853.



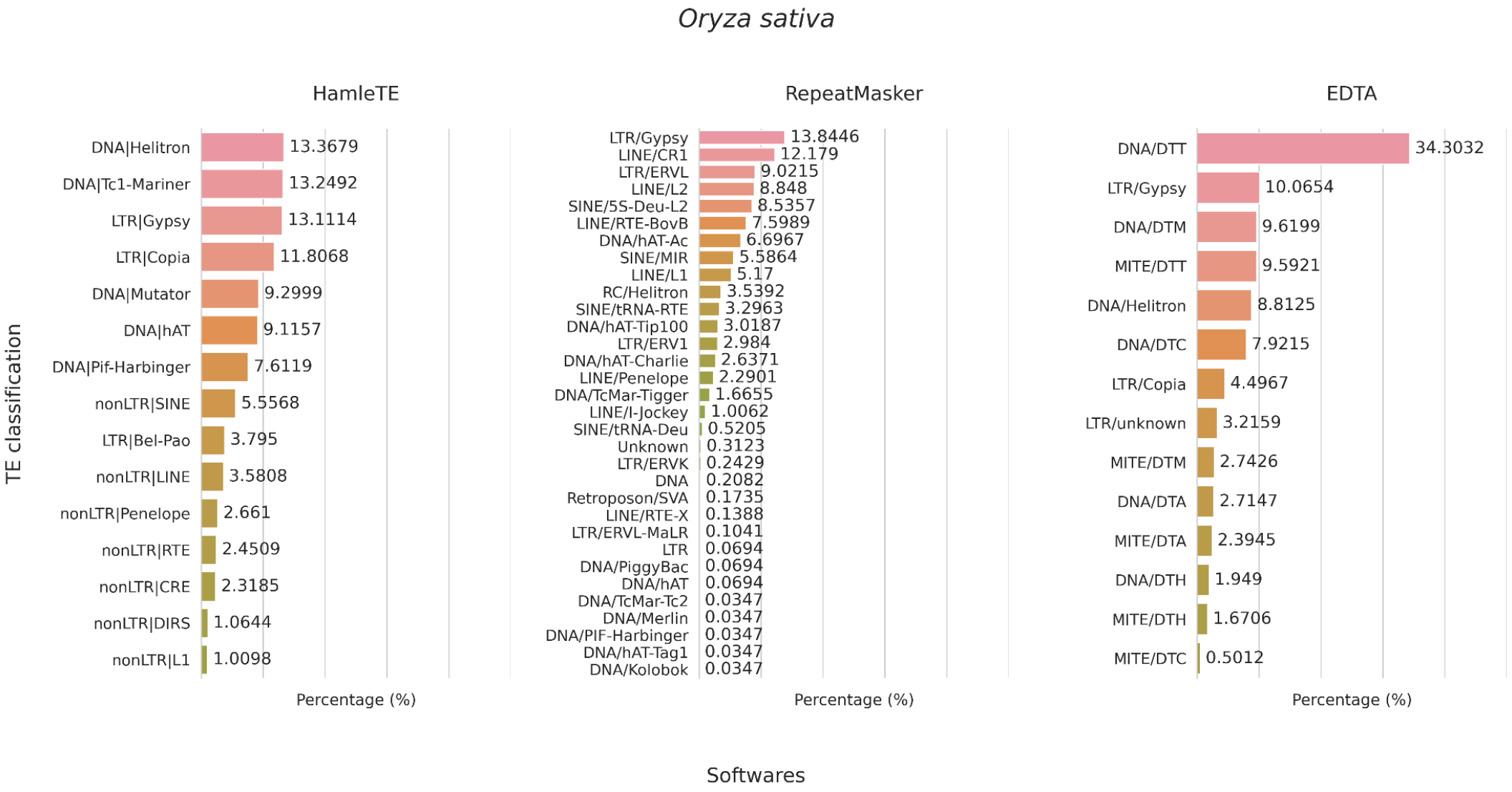
Supplementary Figure 7. Model 6 training and validation accuracy across epochs to learn how to classify non-LTR elements at superfamily level. Training accuracy score was 0.916 and validation accuracy had a score of 0.886.



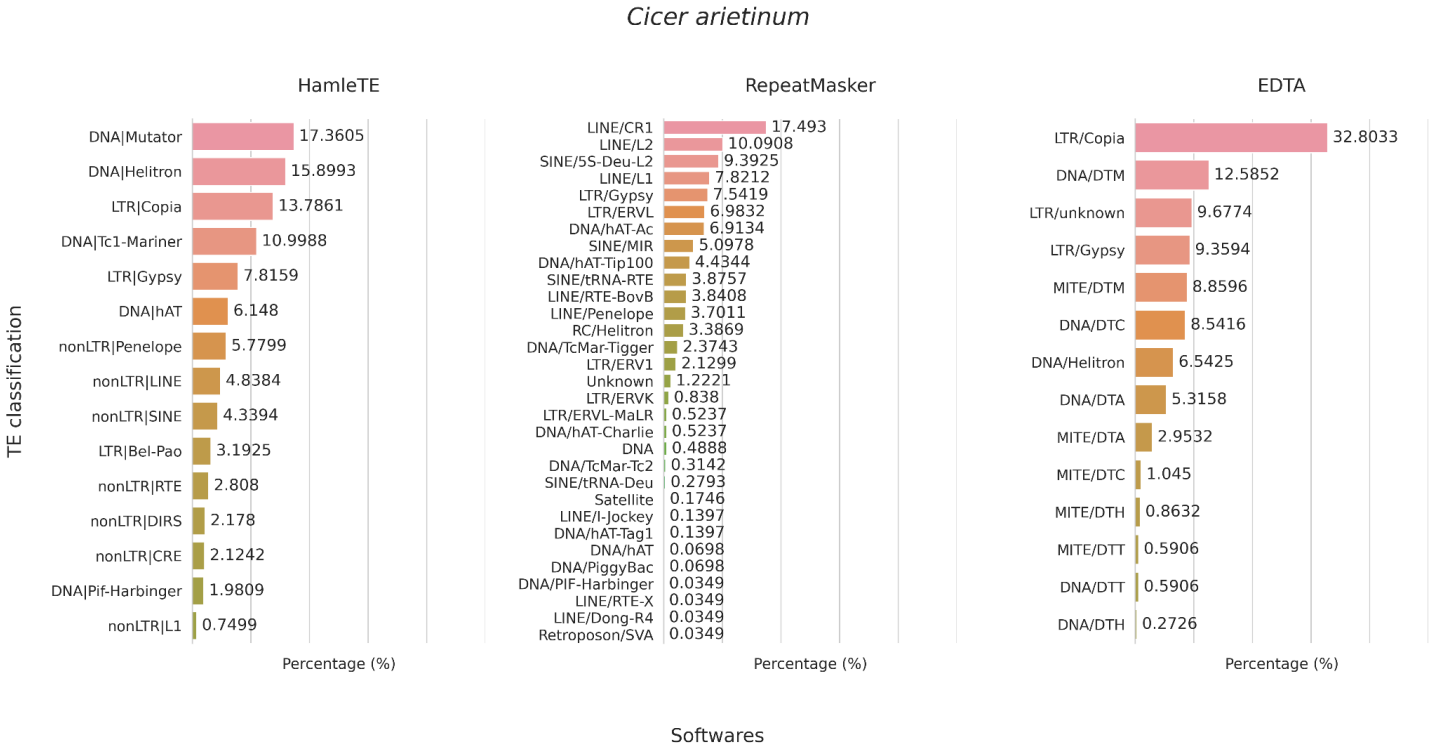
Supplementary figure 8. Total number of sequences in the TE library generated by EDTA, HamleTE e RepeatMasker for seven genomes. HamleTE's workflow in annotation mode found the highest number of sequences classified as TEs in the genomes analyzed when compared to EDTA and RepeatMasker.



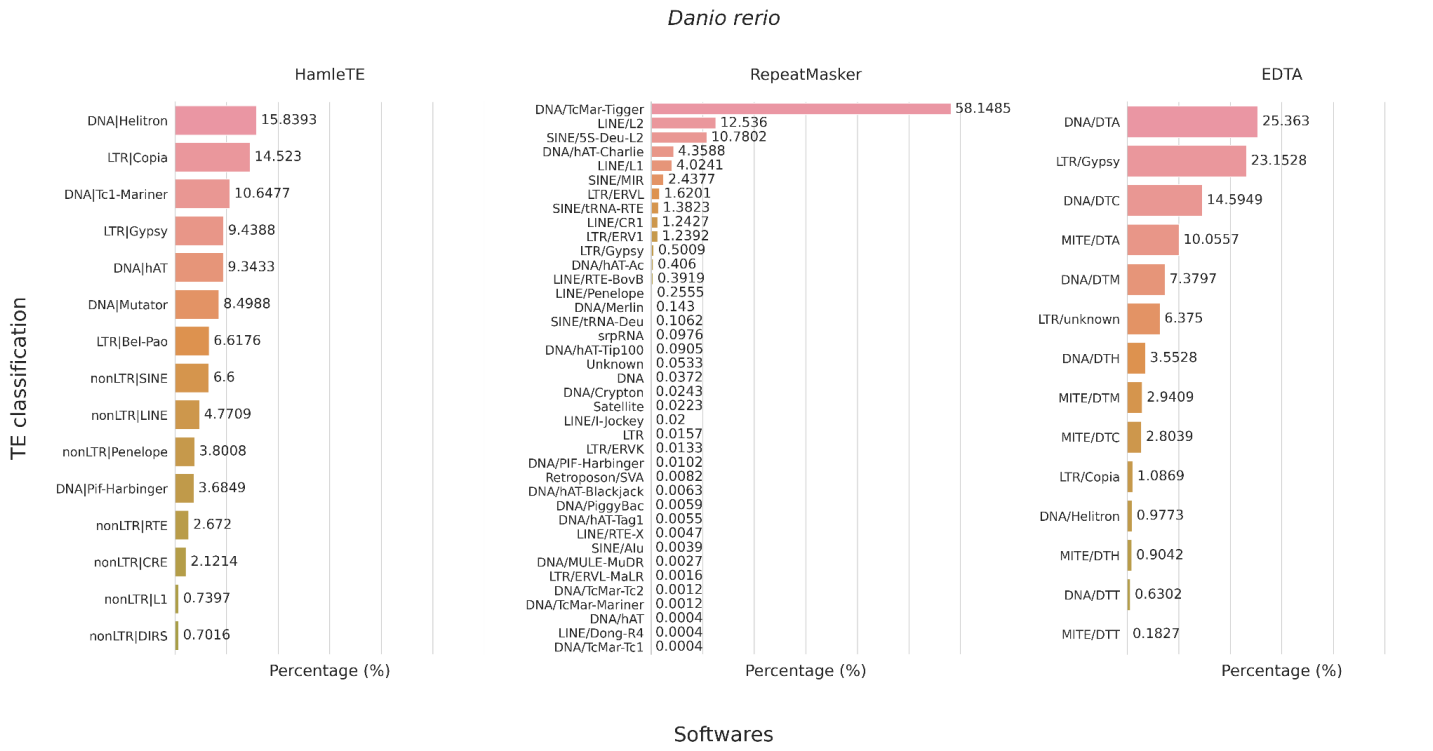
Supplementary figure 9. TEs distribution in percentage of sequences for the *D. melanogaster* genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.



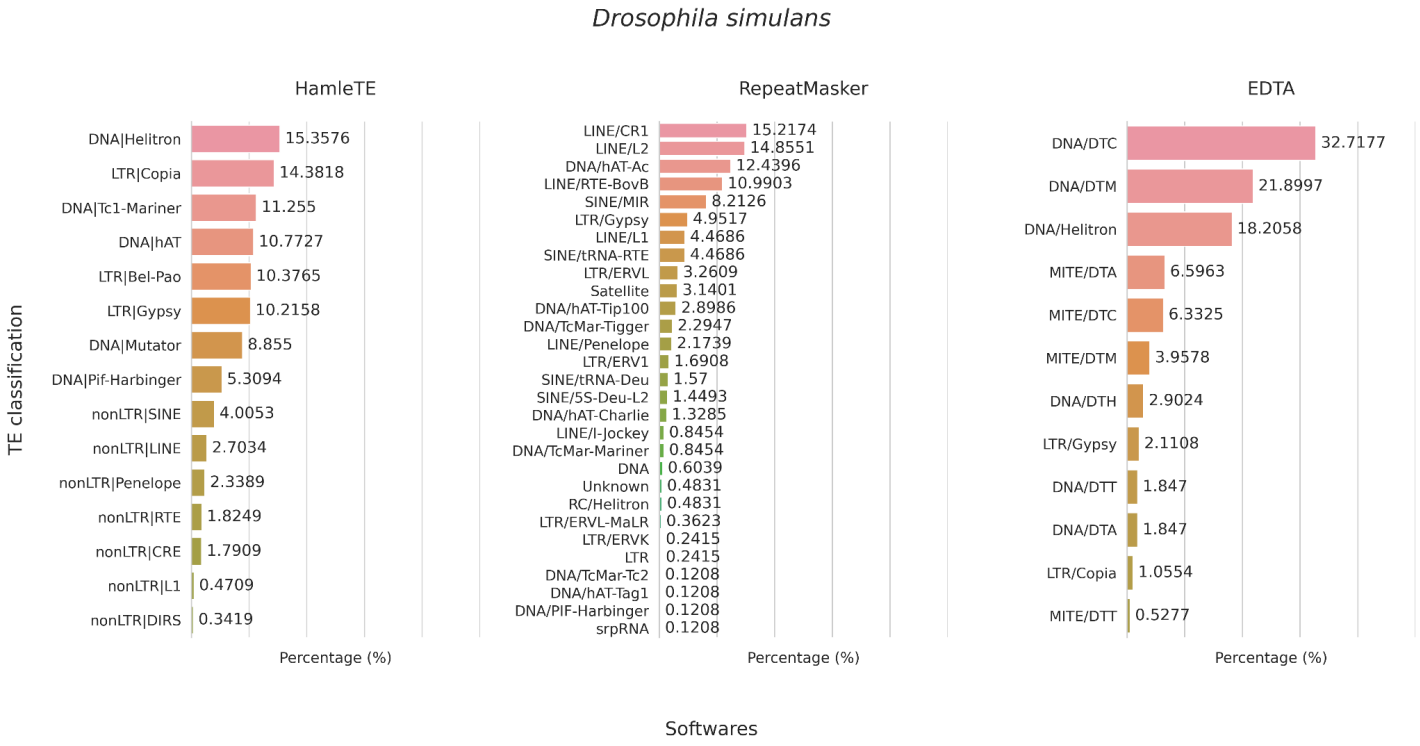
Supplementary figure 10. TEs distribution in percentage of sequences for the *O. sativa* genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.



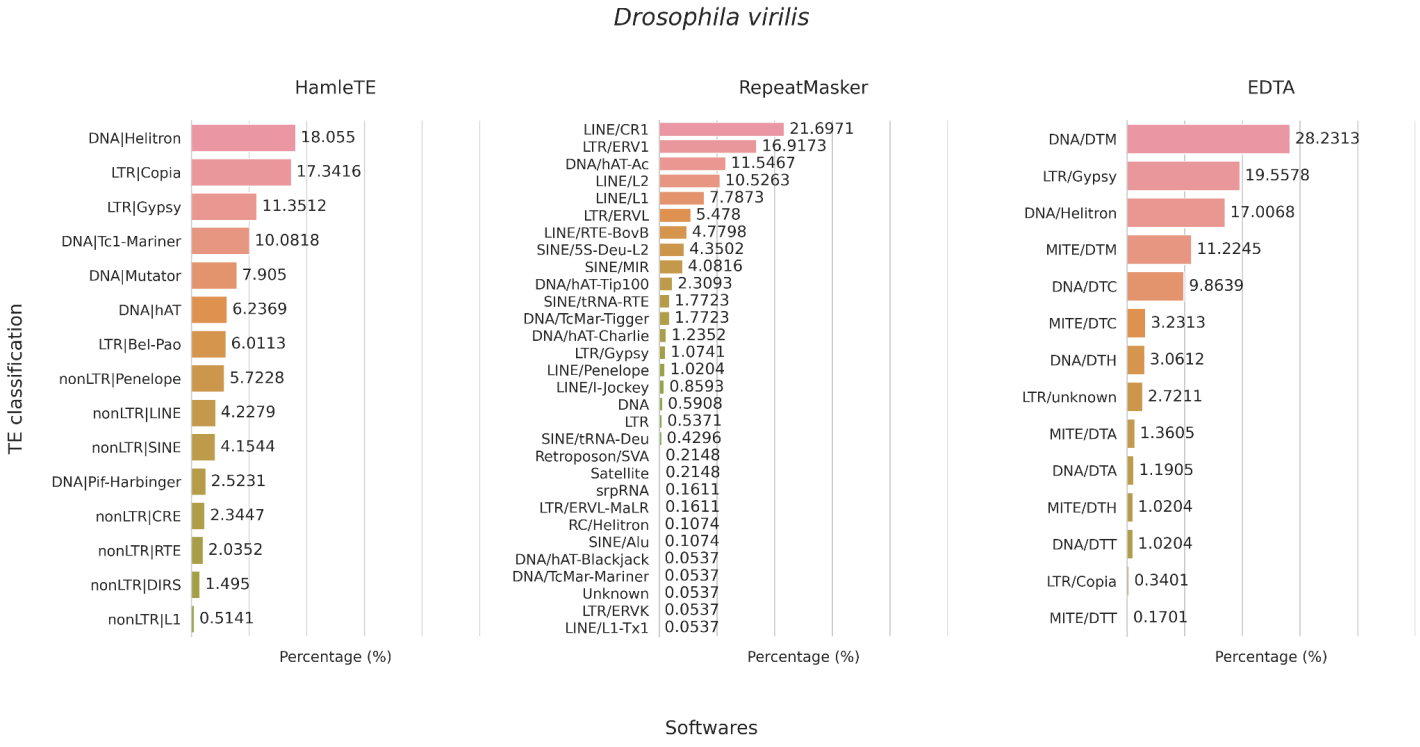
Supplementary figure 11. TEs distribution in percentage of sequences for the *C. arietinum* genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.



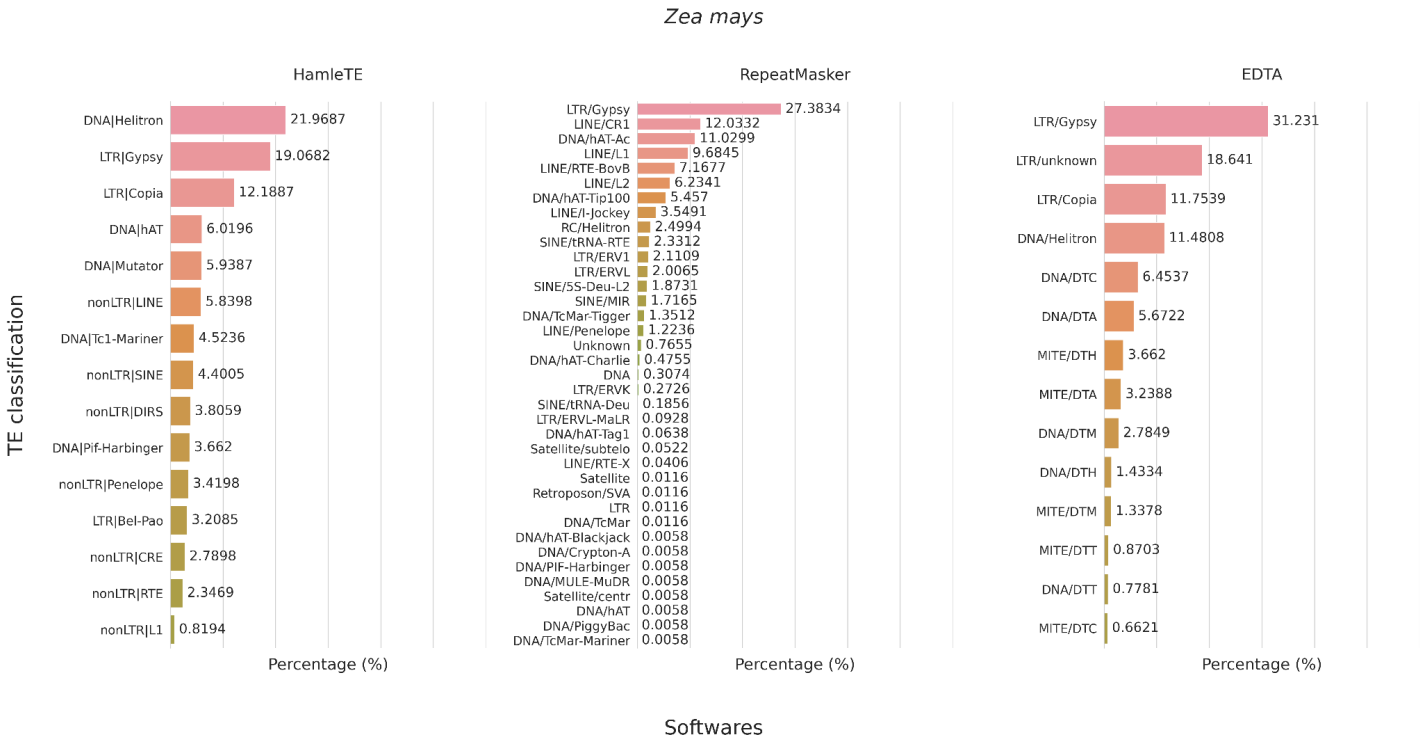
Supplementary figure 12. TEs distribution in percentage of sequences for the *D. rerio* genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.



Supplementary figure 13. TEs distribution in percentage of sequences for the *D. simulans* genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.

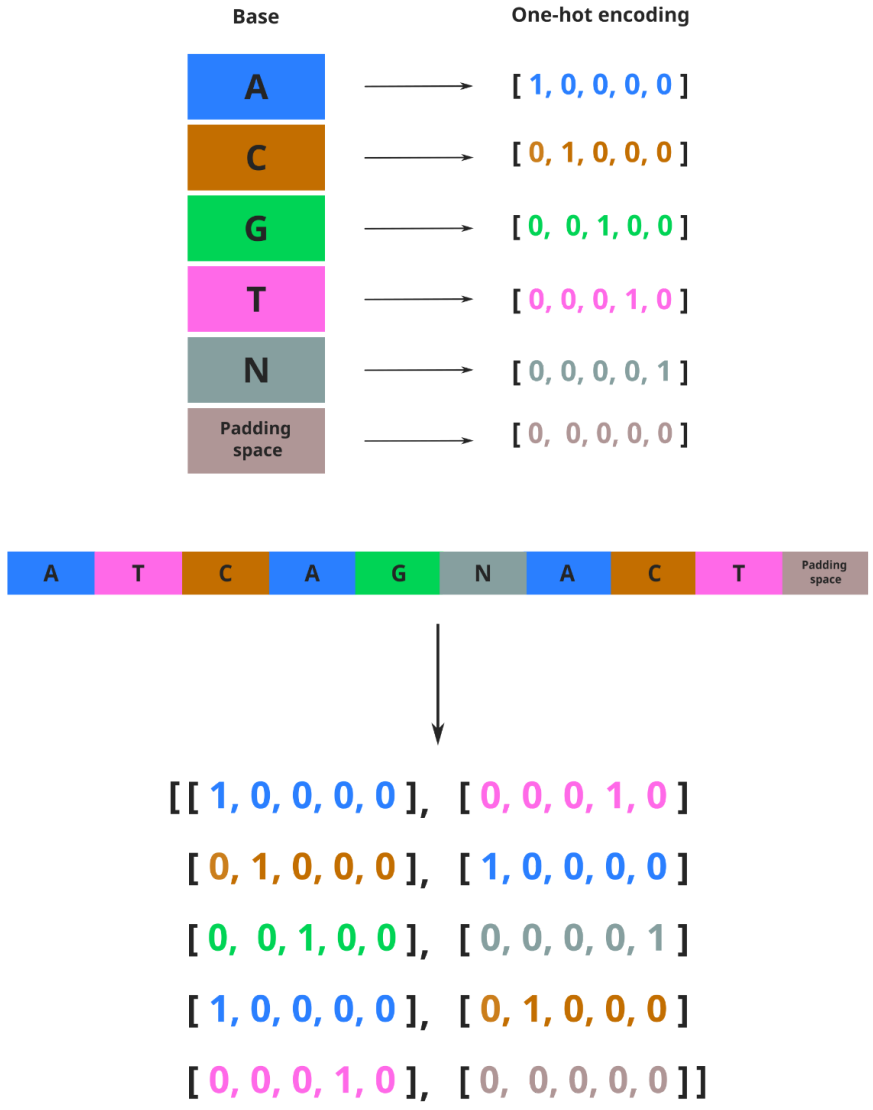


Supplementary figure 14. TEs distribution in percentage of sequences for the *D. virilis* genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.

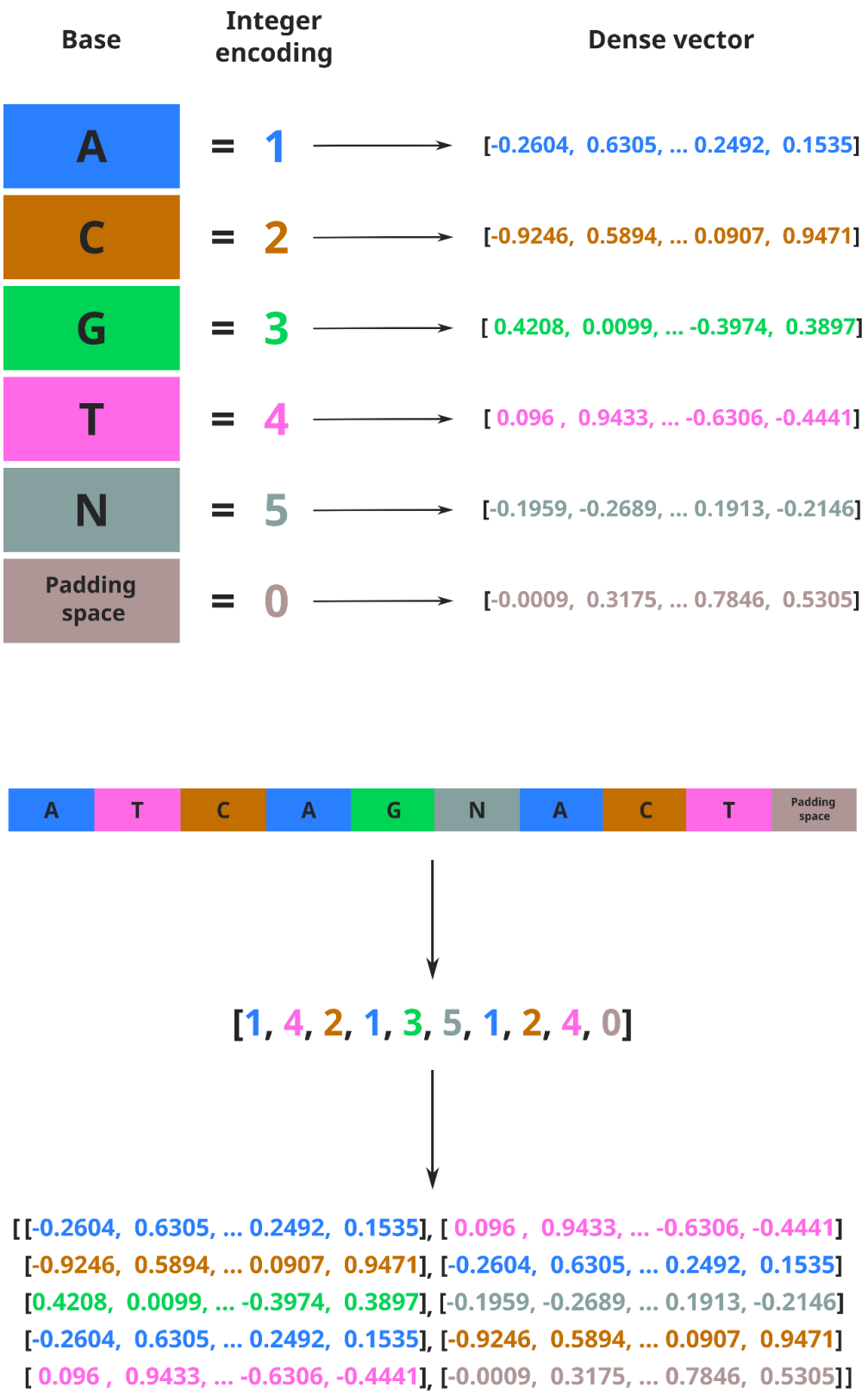


Supplementary figure 15. TEs distribution in percentage of sequences for the Z. mays genome. It shows the percentage found for TE superfamilies in the libraries generated by HamleTE, RepeatMasker and EDTA.

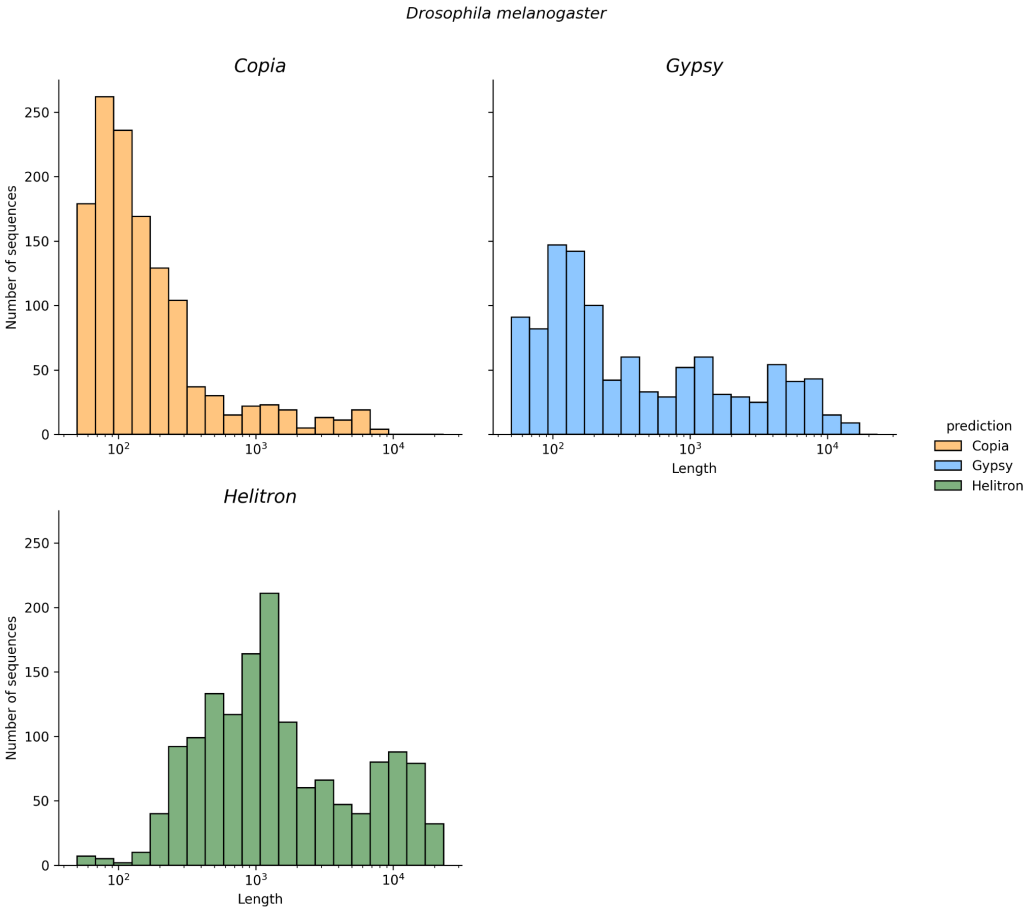
	A	C	G	T	N
A	1	0	0	0	0
C	0	1	0	0	0
G	0	0	1	0	0
T	0	0	0	1	0
N	0	0	0	0	1



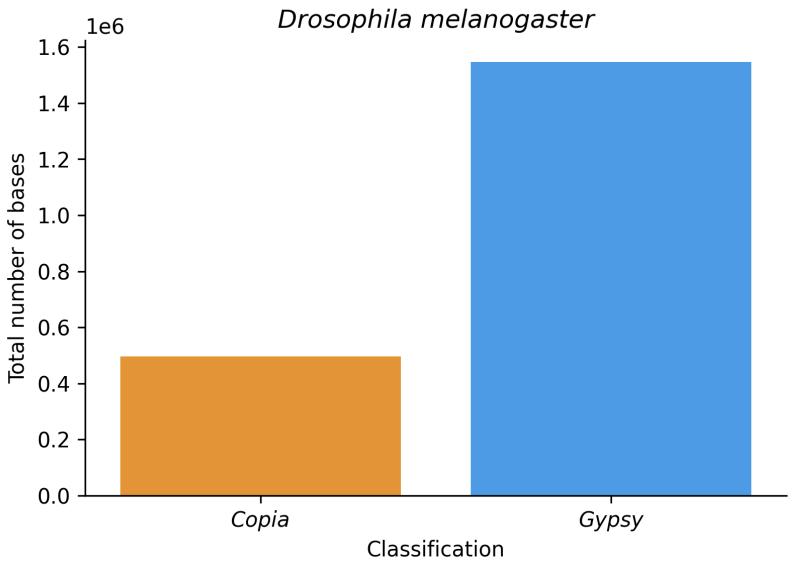
Supplementary figure 16. Schematic representation of applying the one-hot encoding method to nucleotide sequences. Each nucleotide is represented by a sparse vector. The whole sequence is then represented by a 2-dimensional sparse vector.



Supplementary figure 17. Schematic representation of applying the embedding method to nucleotide sequences. Each nucleotide is represented by a dense vector. The whole sequence is then represented by a 2-dimensional dense vector that captures the semantic relationships for sequences of a certain category.



Supplementary figure 18. Size distribution of the sequences classified as *Copia*, *Gypsy* and *Helitron* in the TE library generated by HamleTE for the *D. melanogaster* genome. Most *Copia* sequences are have length in the range of 100 to 400 bases, contrasting with a more harmonic distribution size-wise for *Gypsy* and *Helitron*.



Supplementary figure 19. Total number of bases for Copia and Gypsy in the TE library generated by HamleTE for *D. melanogaster*, showing that *Gypsy* is more representative in total number of bases, although having less sequences in the HamleTE's generated library.

CAPÍTULO 4

CONCLUSÕES GERAIS E PERSPECTIVAS

CONCLUSÕES GERAIS E PERSPECTIVAS

Os TEs são entes genéticos peculiares cuja relevância vagou entre cenários de ferrenha negação de sua natureza móvel, passando por seu desapreço ao ser tachado como “junk” DNA (em português comumente traduzido como DNA “lixo”) até, por fim, com o avanço das pesquisas na área genômica, receber o devido reconhecimento como elemento genético essencial para a diversidade biológica por sua ação como agente modificador, abrangendo processos como a própria especiação, e sua ligação na patogênese de diferentes tipos de câncer e enfermidades ligadas ao envelhecimento. Fortuitamente, este avanço, apesar do dilatado espaço temporal, possibilitou a justa premiação em vida à Barbara McClintock pela descoberta dos TEs.

A natureza repetitiva e diversa mesmo entre elementos de mesmos níveis hierárquicos são componentes relacionados a sua difícil identificação em genomas, fatores estes, que certamente influenciaram no ceticismo inicial para com os TEs. Logo, ferramentas e métodos capazes de identificar com confiabilidade os TEs possíveis de se encontrar nos genomas se fazem uma necessidade, seja com o intuito de mascarar sequências repetitivas para a anotação de genes codificantes de proteínas, ou para a identificação de TEs para estudo de sua estrutura e efeitos biológicos. Com esta ideia em mente, o presente trabalho buscou demonstrar o *status* atual das ferramentas de anotação e classificação de TEs, e apresentar uma nova solução, HamleTE, para auxiliar neste mister.

O manuscrito de revisão intitulado “*The good, the bad and the ugly about transposable elements annotation tools*” teve como objetivo apresentar as principais ferramentas para a anotação de TEs, seus pontos fracos e fortes, e, principalmente, instigar sobre a importância dos *softwares* serem democráticos no que concerne sua usabilidade. Um excelente *software*, mas com uso complexo por padrão, torna-se óbice, impedindo usuários menos técnicos a nível computacional de usar a ferramenta em questão de forma adequada ou mesmo de usá-la absolutamente. Cada nova ferramenta desenvolvida objetiva aprimorar o estado-da-arte corrente, contudo, é imprescindível oferecer uma experiência

amigável aos usuários, sendo de simples uso por padrão e rica em recursos quando demandado pelo usuário.

O manuscrito cerne desta tese revoca as particularidades dos TEs e apresenta HamleTE como uma nova alternativa para a anotação de TEs que mescla um *workflow* prático e eficiente de extração de sequências repetitivas, com o poder do *deep learning* de identificação de padrões para as classificar como TEs até um nível de superfamília, algo que mesmo ferramentas consolidadas baseadas em métodos de similaridade e usando *workflows* mais complexos por vezes não são capazes. HamleTE atesta a capacidade do *deep learning* como método auxiliar no estudo de TEs, mesmo com toda a complexidade inerente a estes elementos, em uma ferramenta focada em ser amigável ao usuário sem deixar de lado a performance e confiabilidade dos resultados.

A contribuição do estudo metodológico e comparativo deste trabalho vai além da criação de uma ferramenta. Discute as melhores práticas de métodos de *deep learning* voltadas para o uso nas ciências de estudo da vida. Este ponto remonta ao princípio basilar da bioinformática: aliar métodos oriundos das ciências da computação a fim de melhor compreender fenômenos biológicos. Os métodos de *deep learning* usados no desenvolvimento de inteligência artificial têm avançado cada vez mais nas diferentes áreas do conhecimento e a pesquisa dentro da área da bioinformática tende a acompanhar este progresso. Isto posto, é essencial o desenvolvimento de trabalhos como este na formação de cientistas do campo.

Pensando na construção do conhecimento e desenvolvimento como cientista, este trabalho traz em anexo um estudo sobre a transferência horizontal de bactérias do gênero *Wolbachia* entre hospedeiros de níveis taxonômicos distantes, evento nomeado em inglês como *host shift* (HS). De forma semelhante aos TEs, Wolbachia tem grande impacto nos organismos que as carreiam, causando alterações fisiológicas e fenotípicas, com influência até mesmo na especiação de artrópodes (Gomes et al. 2022). Com o uso de bioinformática, este trabalho buscou trazer luz ao muitas vezes preterido evento de HS de *Wolbachia*

em artrópodes, discutindo sua importância e o comparando dentro deste ramo da árvore da vida.

Ambos os estudos de TEs e HS possuem aspectos convergentes, importantes nos estudos relacionados à evolução em níveis genéticos e moleculares, principalmente no que diz respeito à transmissão e ao impacto de elementos genéticos móveis. O uso de estratégias bioinformáticas mais tradicionais aliado ao estudo e desenvolvimento de ferramentas buscando o estado-da-arte demonstram a abrangência do campo da bioinformática e a relevância de uma formação sólida como pesquisador. Este trabalho foi então capaz de contribuir em diferentes porções dentro do espectro da bioinformática voltada à genética molecular ao levantar discussões pertinentes sobre assuntos fundamentais que podem acabar por não receber a atenção devida, como no caso do trabalho em anexo, e apresentar soluções na fronteira do conhecimento, tal qual abordado no tema principal, com o uso de *deep learning* para resolver problemas complexos como a anotação de TEs.

Por fim, fica aberta a possibilidade de se refinar as ferramentas baseando-se nos métodos apresentados e, a partir destes, desenvolver programas "*data-driven*", isto é, além de uma abordagem generalista, um programa treinável pelo usuário para anotação em níveis mais específicos dentro de gênero ou espécie. Além disso, pode-se tentar desenvolver métodos com o uso de *deep learning* capazes de extrair e classificar TEs sem o uso de ferramentas externas.

REFERÊNCIAS BIBLIOGRÁFICAS

Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Van Esesn BC, Awwal AAS and Asari VK (2018) The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. doi: 10.48550/arXiv.1803.01164

Anderson SN, Stitzer MC, Brohammer AB, Zhou P, Noshay JM, O'Connor CH, Hirsch CD, Ross-Ibarra J, Hirsch CN and Springer NM (2019) Transposable elements contribute to dynamic genome content in maize. Plant J 100:1052–1065.

Andrenacci D, Cavaliere V and Lattanzi G (2020) The role of transposable elements activity in aging and their possible involvement in laminopathic diseases. Ageing Res Rev 57:100995.

Arkhipova IR (2017) Using bioinformatic and phylogenetic approaches to classify transposable elements and understand their complex evolutionary histories. Mob DNA 8:19.

Ashiquzzaman A, Tushar AK, Islam MdR, Shon D, Im K, Park J-H, Lim D-S and Kim J (2018) Reduction of Overfitting in Diabetes Prediction Using Deep Learning Neural Network. In: Kim KJ, Kim H and Baek N (eds) IT Convergence and Security 2017. Springer, Singapore, pp 35–43

Baldi P, Sadowski P and Lu Z (2018) Learning in the machine: Random backpropagation and the deep learning channel. Artif Intell 260:1–35.

Batzer MA and Deininger PL (2002) Alu repeats and human genomic diversity. Nat Rev Genet 3:370–379.

Bejani MM and Ghatee M (2021) A systematic review on overfitting control in shallow and deep neural networks. Artif Intell Rev 54:6391–6438.

Biémont C (2010) A Brief History of the Status of Transposable Elements: From Junk DNA to Major Players in Evolution. Genetics 186:1085–1093.

Blasco T, Sánchez JS and Garcia V (2023) A Survey of Uncertainty Quantification in Deep Learning for Financial Time Series Prediction. doi: 10.2139/ssrn.4506769

Bourque G, Burns KH, Gehring M, Gorbunova V, Seluanov A, Hammell M, Imbeault M, Izsvák Z, Levin HL, Macfarlan TS et al. (2018) Ten things you should know about transposable elements. Genome Biol 19:199.

Britten RJ (2010) Transposable element insertions have strongly affected human evolution. Proc Natl Acad Sci 107:19945–19948.

Broecker F and Moelling K (2019) Evolution of Immune Systems From Viruses and Transposable Elements. Front Microbiol 10:51.

Burns KH (2017) Transposable elements in cancer. Nat Rev Cancer 17:415–424.

Campbell M, Hoane AJ and Hsu F (2002) Deep Blue. Artif Intell 134:57–83.

Canapa A, Barucca M, Biscotti MA, Forconi M and Olmo E (2016) Transposons, Genome Size, and Evolutionary Insights in Animals. Cytogenet Genome Res 147:217–239.

Cancian M, da Fontoura Gomes TMF and Loreto ELS (2022) Somatic Mobilization: High Somatic Insertion Rate of mariner Transposable Element in Drosophila simulans. Insects 13:454.

Cao C, Liu F, Tan H, Song D, Shu W, Li W, Zhou Y, Bo X and Xie Z (2018) Deep Learning and Its Applications in Biomedicine. Genomics Proteomics Bioinformatics 16:17–32.

Caruana R and Niculescu-Mizil A (2006) An empirical comparison of supervised

learning algorithms. Proceedings of the 23rd international conference on Machine learning. Association for Computing Machinery, New York, NY, USA, pp 161–168
Celebi ME and Aydin K (2016) Unsupervised Learning Algorithms. doi: 10.1007/978-3-319-24211-8
Chandra B and Sharma RK (2016) Deep learning with adaptive learning rate using laplacian score. Expert Syst Appl 63:1–7.
Chen H, Xiong F, Wu D, Zheng L, Peng A, Hong X, Tang B, Lu H, Shi H and Zheng H (2017) Assessing impacts of data volume and data set balance in using deep learning approach to human activity recognition. 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). pp 1160–1165
Ching T, Zhu X and Garmire LX (2018) Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data. PLOS Comput Biol 14:e1006076.
Choi D, Shallue CJ, Nado Z, Lee J, Maddison CJ and Dahl GE (2020) On Empirical Comparisons of Optimizers for Deep Learning. doi: 10.48550/arXiv.1910.05446
Cowan JD (1990) Discussion: McCulloch-Pitts and related neural nets from 1943 to 1989. Bull Math Biol 52:73–97.
Cui Y, Jia M, Lin T-Y, Song Y and Belongie S (2019) Class-Balanced Loss Based on Effective Number of Samples. doi: 10.48550/arXiv.1901.05555
de Boer P-T, Kroese DP, Mannor S and Rubinstein RY (2005) A Tutorial on the Cross-Entropy Method. Ann Oper Res 134:19–67.
Deneweth J, Van de Peer Y and Vermeirssen V (2022) Nearby transposable elements impact plant stress gene regulatory networks: a meta-analysis in A. thaliana and S. lycopersicum. BMC Genomics 23:18.
Denil M, Shakibi B, Dinh L, Ranzato MA and de Freitas N (2013) Predicting Parameters in Deep Learning. Adv. Neural Inf. Process. Syst. 26:
Di Stefano L (2022) All Quiet on the TE Front? The Role of Chromatin in Transposable Element Silencing. Cells 11:2501.
Dong S, Wang P and Abbas K (2021) A survey on deep learning and its applications. Comput Sci Rev 40:100379.
Doolittle WF (2022) All about levels: transposable elements as selfish DNAs and drivers of evolution. Biol Philos 37:24.
Doolittle WF and Brunet TDP (2017) On causal roles and selected effects: our genome is mostly junk. BMC Biol 15:116.
El-Amir H and Hamdy M (2020) Deep Learning Fundamentals. In: El-Amir H and Hamdy M (eds) Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow. Apress, Berkeley, CA, pp 279–343
Finnegan DJ (1989) Eukaryotic transposable elements and genome evolution. Trends Genet TIG 5:103–107.
Floridi L (2020) AI and Its New Winter: from Myths to Realities. Philos Technol 33:1–3.
François-Lavet V, Henderson P, Islam R, Bellemare MG and Pineau J (2018) An Introduction to Deep Reinforcement Learning. Found Trends® Mach Learn 11:219–354.
Galbraith JD and Hayward A (2023) The influence of transposable elements on animal colouration. Trends Genet. doi: 10.1016/j.tig.2023.04.005
Gao L, Zhang L, Liu C and Wu S (2020) Handling Imbalanced Medical Image

Data: A Deep-Learning-Based One-Class Classification Approach. Artif Intell Med 108:101935.
García-Ordás MT, Benavides C, Benítez-Andrades JA, Alaiz-Moretón H and García-Rodríguez I (2021) Diabetes detection using deep learning techniques with oversampling and feature augmentation. Comput Methods Programs Biomed 202:105968.
Gavrilov AD, Jordache A, Vasdani M and Deng J (2018) Preventing Model Overfitting and Underfitting in Convolutional Neural Networks. Int J Softw Sci Comput Intell IJSSCI 10:19–28.
Ghojogh B and Crowley M (2023) The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. doi: 10.48550/arXiv.1905.12787
Gomes TMFF, Wallau GL and Loreto ELS (2022) Multiple long-range host shifts of major Wolbachia supergroups infecting arthropods. Sci Rep 12:8131.
Gordon-Rodriguez E, Loaiza-Ganem G, Pleiss G and Cunningham JP (2020) Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning. PMLR, pp 1–10
Han W, Zhang Z, Zhang Y, Yu J, Chiu C-C, Qin J, Gulati A, Pang R and Wu Y (2020) ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context. doi: 10.48550/arXiv.2005.03191
Hayward A and Gilbert C (2022) Transposable elements. Curr Biol 32:R904–R909.
Hedges DJ and Deininger PL (2007) Inviting Instability: Transposable elements, Double-strand breaks, and the Maintenance of Genome Integrity. Mutat Res 616:46–59.
Hendler J (2008) Avoiding Another AI Winter. IEEE Intell Syst 23:2–4.
Hernandez J, Carrasco-Ochoa JA and Martínez-Trinidad JF (2013) An Empirical Study of Oversampling and Undersampling for Instance Selection Methods on Imbalance Datasets. In: Ruiz-Shulcloper J and Sanniti di Baja G (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer, Berlin, Heidelberg, pp 262–269
Hinton GE and Salakhutdinov RR (2006) Reducing the Dimensionality of Data with Neural Networks. Science 313:504–507.
Hirsch CD and Springer NM (2017) Transposable element influences on gene expression in plants. Biochim Biophys Acta BBA - Gene Regul Mech 1860:157–165.
Hsu F-H (1999) IBM's Deep Blue Chess grandmaster chips. IEEE Micro 19:70–81.
Ivancevic A and Chuong EB (2020) Transposable elements teach T cells new tricks. Proc Natl Acad Sci 117:9145–9147.
Jakhar D and Kaur I (2020) Artificial intelligence, machine learning and deep learning: definitions and differences. Clin Exp Dermatol 45:131–132.
Johnson JM and Khoshgoftaar TM (2019a) Survey on deep learning with class imbalance. J Big Data 6:27.
Johnson JM and Khoshgoftaar TM (2019b) Deep Learning and Data Sampling with Imbalanced Big Data. 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI). pp 175–183
Johny A and Madhusoodanan KN (2021) Dynamic Learning Rate in Deep CNN Model for Metastasis Detection and Classification of Histopathology Images. Comput Math Methods Med 2021:e5557168.
Kaelbling LP, Littman ML and Moore AW (1996) Reinforcement Learning: A

Survey. J Artif Intell Res 4:237–285.
Kanal LN (2003) Perceptron. Encyclopedia of Computer Science. John Wiley and Sons Ltd., GBR, pp 1383–1385
Kidwell MG (2002) Transposable elements and the evolution of genome size in eukaryotes. Genetica 115:49–63.
Kolluri J, Kotte VK, Phridviraj MSB and Razia S (2020) Reducing Overfitting Problem in Machine Learning Using Novel L1/4 Regularization Method. 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184). pp 934–938
Krawczyk B, Woźniak M and Herrera F (2014) Weighted one-class classification for different types of minority class examples in imbalanced data. 2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). pp 337–344
Kussul E, Baidyk T, Kasatkina L and Lukovich V (2001) Rosenblatt perceptrons for handwritten digit recognition. IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222). pp 1516–1520 vol.2
Lanciano S and Cristofari G (2020) Measuring and interpreting transposable element expression. Nat Rev Genet 21:721–736.
Lashgari E, Liang D and Maoz U (2020) Data augmentation for deep-learning-based electroencephalography. J Neurosci Methods 346:108885.
LeCun Y, Bengio Y and Hinton G (2015) Deep learning. Nature 521:436–444.
Li Q, Yan M and Xu J (2021) Optimizing Convolutional Neural Network Performance by Mitigating Underfitting and Overfitting. 2021 IEEE/ACIS 19th International Conference on Computer and Information Science (ICIS). pp 126–131
Li Y (2018) Deep Reinforcement Learning: An Overview. doi: 10.48550/arXiv.1701.07274
Li Y, Huang X, Huang X, Gao X, Hu R, Yang X and He Y-L (2023) Machine learning and multilayer perceptron enhanced CFD approach for improving design on latent heat storage tank. Appl Energy 347:121458.
Lillicrap TP, Santoro A, Marris L, Akerman CJ and Hinton G (2020) Backpropagation and the brain. Nat Rev Neurosci 21:335–346.
Liu B, Blekas K and Tsoumakas G (2022) Multi-label sampling based on local label imbalance. Pattern Recognit 122:108294.
Liu Z, Xu Z, Jin J, Shen Z and Darrell T (2023) Dropout Reduces Underfitting. doi: 10.48550/arXiv.2303.01500
Luo H, Xiong C, Fang W, Love PED, Zhang B and Ouyang X (2018) Convolutional neural networks: Computer vision-based workforce activity assessment in construction. Autom Constr 94:282–289.
Martorell-Marugán J, Tabik S, Benhammou Y, Val C del, Zwir I, Herrera F and Carmona-Sáez P (2019) Deep Learning in Omics Data Analysis and Precision Medicine. Exon Publ 37–53.
McClintock B (1947) Mutable loci in maize. Mutable Loci Maize
McClintock B (1956) Controlling Elements and the Gene. Cold Spring Harb Symp Quant Biol 21:197–216.
McCulloch WS and Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133.
Meena D, Behera BK, Das P, Prusty AK, Kumar S and MEENA M (2012) Transposable elements: Strategies and mechanism of transposition in Danio rerio,

a genetic model. J Bio-Sci 7:223–229.
Mehedi Shamrat FMJ, Jubair MdA, Billah MdM, Chakraborty S, Alauddin Md and Ranjan R (2021) A Deep Learning Approach for Face Detection using Max Pooling. 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI). pp 760–764
Mombach DM, da Fontoura Gomes TMF and Loreto ELS (2022a) Stress does not induce a general transcription of transposable elements in Drosophila. Mol Biol Rep 49:9033–9040.
Mombach DM, Fontoura Gomes TMF da, Silva MM and Loreto ÉLS (2022b) Molecular and biological effects of Cisplatin in Drosophila. Comp Biochem Physiol Part C Toxicol Pharmacol 252:109229.
Murtagh F (1991) Multilayer perceptrons for classification and regression. Neurocomputing 2:183–197.
Nakamura M, Köhler C and Hennig L (2019) Tissue-specific transposon-associated small RNAs in the gymnosperm tree, Norway spruce. BMC Genomics 20:997.
Ochmann MT and Ivics Z (2021) Jumping Ahead with Sleeping Beauty: Mechanistic Insights into Cut-and-Paste Transposition. Viruses 13:76.
Oggenfuss U, Badet T, Wicker T, Hartmann FE, Singh NK, Abraham L, Karisto P, Vonlanthen T, Mundt C, McDonald BA et al. (2021) A population-level invasion by transposable elements triggers genome expansion in a fungal pathogen. eLife 10:e69249.
Ohno S (1972) So much “junk” DNA in our genome. Brookhaven Symp Biol 23:366–370.
Okewu E, Adewole P and Sennaike O (2019) Experimental Comparison of Stochastic Optimizers in Deep Learning. In: Misra S, Gervasi O, Murgante B, Stankova E, Korkhov V, Torre C, Rocha AMAC, Taniar D, Apduhan BO and Tarantino E (eds) Computational Science and Its Applications – ICCSA 2019. Springer International Publishing, Cham, pp 704–715
O’Shea K and Nash R (2015) An Introduction to Convolutional Neural Networks. doi: 10.48550/arXiv.1511.08458
Oubounyt M, Louadi Z, Tayara H and Chong KT (2019) DeePromoter: Robust Promoter Predictor Using Deep Learning. Front. Genet. 10:
Petersen M, Armisén D, Gibbs RA, Hering L, Khila A, Mayer G, Richards S, Niehuis O and Misof B (2019) Diversity and evolution of the transposable element repertoire in arthropods with particular reference to insects. BMC Ecol Evol 19:11.
Piégu B, Bire S, Arensburger P and Bigot Y (2015) A survey of transposable element classification systems – A call for a fundamental update to meet the challenge of their diversity and complexity. Mol Phylogenet Evol 86:90–109.
Ravindran S (2012) Barbara McClintock and the discovery of jumping genes. Proc Natl Acad Sci 109:20198–20199.
Rawat W and Wang Z (2017) Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Comput 29:2352–2449.
Rice L, Wong E and Kolter Z (2020) Overfitting in adversarially robust deep learning. Proceedings of the 37th International Conference on Machine Learning. PMLR, pp 8093–8104
Rojas R (1996) The Backpropagation Algorithm. In: Rojas R (ed) Neural Networks: A Systematic Introduction. Springer, Berlin, Heidelberg, pp 149–182

Rosenblatt F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychol Rev 65:386–408.
Rosenblatt F (1962) Principles of neurodynamics. Perceptrons and the theory of brain mechanisms.
Roy Y, Banville H, Albuquerque I, Gramfort A, Falk TH and Faubert J (2019) Deep learning-based electroencephalography analysis: a systematic review. J Neural Eng 16:051001.
Saini M and Susan S (2022) Diabetic retinopathy screening using deep learning for multi-class imbalanced datasets. Comput Biol Med 149:105989.
Salakhutdinov R and Hinton G (2007) Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics. PMLR, pp 412–419
Saleh A, Macia A and Muotri AR (2019) Transposable Elements, Inflammation, and Neurological Disease. Front. Neurol. 10:
Saravanan R and Sujatha P (2018) A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). pp 945–949
Schmidhuber J (2015) Deep learning in neural networks: An overview. Neural Netw 61:85–117.
Schmidt RM, Schneider F and Hennig P (2021) Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers. Proceedings of the 38th International Conference on Machine Learning. PMLR, pp 9367–9376.
Schrader L and Schmitz J (2019) The impact of transposable elements in adaptive evolution. Mol Ecol 28:1537–1549.
Seising R (2018) The Emergence of Fuzzy Sets in the Decade of the Perceptron—Lotfi A. Zadeh's and Frank Rosenblatt's Research Work on Pattern Classification. Mathematics 6:110.
Serrato-Capuchina A and Matute DR (2018) The Role of Transposable Elements in Speciation. Genes 9:254.
Shahinfar S, Meek P and Falzon G (2020) "How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. Ecol Inform 57:101085.
Shankar K, Zhang Y, Liu Y, Wu L and Chen C-H (2020) Hyperparameter Tuning Deep Learning for Diabetic Retinopathy Fundus Image Classification. IEEE Access 8:118164–118173.
Shao F, Han M and Peng Z (2019) Evolution and diversity of transposable elements in fish genomes. Sci Rep 9:15399.
Sharma N, Sharma R and Jindal N (2021) Machine Learning and Deep Learning Applications-A Vision. Glob Transit Proc 2:24–28.
Shinde PP and Shah S (2018) A Review of Machine Learning and Deep Learning Applications. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). pp 1–6
Sinaga KP and Yang M-S (2020) Unsupervised K-Means Clustering Algorithm. IEEE Access 8:80716–80727.
Skipper K, Andersen P, Sharma N and Mikkelsen J (2013) DNA transposon-based

gene vehicles - Scenes from an evolutionary drive. J Biomed Sci 20:92.
Somuncuoğlu AN, Purutçuoğlu V, Ari F and Gökçay D (2020) Investigation on the Use of Hidden Layers, Different Numbers of Neurons and Different Activation Functions to Detect Pupil Dilation Responses to Stress. 2020 Medical Technologies Congress (TIPTEKNO). pp 1–4
Sotero-Caio CG, Platt RN II, Suh A and Ray DA (2017) Evolution and Diversity of Transposable Elements in Vertebrate Genomes. Genome Biol Evol 9:161–177.
Stitzer MC, Anderson SN, Springer NM and Ross-Ibarra J (2021) The genomic ecosystem of transposable elements in maize. PLoS Genet 17:e1009768.
Suk H-I (2017) Chapter 1 - An Introduction to Neural Networks and Deep Learning. In: Zhou SK, Greenspan H and Shen D (eds) Deep Learning for Medical Image Analysis. Academic Press, pp 3–24
Tao Z, XiaoYu C, HuiLing L, XinYu Y, YunCan L and XiaoMin Z (2022) Pooling Operations in Deep Learning: From “Invariable” to “Variable.” BioMed Res Int 2022:e4067581.
Tarekegn AN, Giacobini M and Michalak K (2021) A review of methods for imbalanced multi-label classification. Pattern Recognit 118:107965.
Toosi A, Bottino AG, Saboury B, Siegel E and Rahmim A (2021) A Brief History of AI: How to Prevent Another Winter (A Critical Review). PET Clin 16:449–469.
Uzair M and Jamil N (2020) Effects of Hidden Layers on the Efficiency of Neural networks. 2020 IEEE 23rd International Multitopic Conference (INMIC). pp 1–6
Wang H, Cui Z, Chen Y, Avidan M, Abdallah AB and Kronzer A (2018) Predicting Hospital Readmission via Cost-Sensitive Deep Learning. IEEE/ACM Trans Comput Biol Bioinform 15:1968–1978.
Wang T and Huang P (2023) Superiority of a Convolutional Neural Network Model over Dynamical Models in Predicting Central Pacific ENSO. Adv Atmospheric Sci. doi: 10.1007/s00376-023-3001-1
Wang W and Gang J (2018) Application of Convolutional Neural Network in Natural Language Processing. 2018 International Conference on Information Systems and Computer Aided Education (ICISCAE). pp 64–70
Warren IA, Naville M, Chalopin D, Levin P, Berger CS, Galiana D and Volff J-N (2015) Evolutionary impact of transposable elements on genomic diversity and lineage-specific innovation in vertebrates. Chromosome Res 23:505–531.
Wells JN and Feschotte C (2020) A Field Guide to Eukaryotic Transposable Elements. Annu Rev Genet 54:539–561.
Wessler SR (2006) Transposable elements and the evolution of eukaryotic genomes. Proc Natl Acad Sci 103:17600–17601.
Wicker T, Sabot F, Hua-Van A, Bennetzen JL, Capy P, Chalhoub B, Flavell A, Leroy P, Morgante M, Panaud O et al. (2007) A unified classification system for eukaryotic transposable elements. Nat Rev Genet 8:973–982.
Wythoff BJ (1993) Backpropagation neural networks: A tutorial. Chemom Intell Lab Syst 18:115–155.
Xia B, Zhang W, Wudzinska A, Huang E, Brosh R, Pour M, Miller A, Dasen JS, Maurano MT, Kim SY et al. (2021) The genetic basis of tail-loss evolution in humans and apes. 2021.09.14.460388.
Yedida R and Menzies T (2022) On the Value of Oversampling for Deep Learning in Software Defect Prediction. IEEE Trans Softw Eng 48:3103–3116.
Yekrangi M and Nikolov NS (2023) Domain-Specific Sentiment Analysis: An

Optimized Deep Learning Approach for the Financial Markets. IEEE Access 11:70248–70262.
Young SR, Rose DC, Karnowski TP, Lim S-H and Patton RM (2015) Optimizing deep learning hyper-parameters through an evolutionary algorithm. Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. Association for Computing Machinery, New York, NY, USA, pp 1–5
Zempleni J, Chew YC, Bao B, Pestinger V and Wijeratne SSK (2009) Repression of Transposable Elements by Histone Biotinylation. J Nutr 139:2389–2392.
Zhang L, Yan L, Jiang J, Wang Y, Jiang Y, Yan T and Cao Y (2014) The structure and retrotransposition mechanism of LTR-retrotransposons in the asexual yeast Candida albicans. Virulence 5:655–664.
Zou J, Huss M, Abid A, Mohammadi P, Torkamani A and Telenti A (2019) A primer on deep learning in genomics. Nat Genet 51:12–18.

ANEXO

Artigo - Multiple long-range host shifts of major *Wolbachia* supergroups infecting arthropods

Tiago M. F. F. Gomes, Gabriel L. Wallau & Elgion L. S. Loreto

Submetido: 12 de outubro de 2021
Aceito: 09 de maio de 2022
Publicado: 17 de maio de 2022

Scientific Reports volume 12, Article number: 8131 (2022).
DOI: https://doi.org/10.1038/s41598-022-12299-x

scientific reports

Check for updates

OPEN

Multiple long-range host shifts of major *Wolbachia* supergroups infecting arthropods

Tiago M. F. F. Gomes[1], Gabriel L. Wallau[2] & Elgion L. S. Loreto[1,2,3]✉

Wolbachia **is a genus of intracellular bacterial endosymbionts found in 20–66% of all insect species and a range of other invertebrates. It is classified as a single species, *Wolbachia pipientis*, divided into supergroups A to U, with supergroups A and B infecting arthropods exclusively. *Wolbachia* is transmitted mainly via vertical transmission through female oocytes, but can also be transmitted across different taxa by host shift (HS): the direct transmission of *Wolbachia* cells between organisms without involving vertically transmitted gametic cells. To assess the HS contribution, we recovered 50 orthologous genes from over 1000 *Wolbachia* genomes, reconstructed their phylogeny and calculated gene similarity. Of 15 supergroup A *Wolbachia* lineages, 10 have similarities ranging from 95 to 99.9%, while their hosts' similarities are around 60 to 80%. For supergroup B, four out of eight lineages, which infect diverse and distantly-related organisms such as Acari, Hemiptera and Diptera, showed similarities from 93 to 97%. These results show that *Wolbachia* genomes have a much higher similarity when compared to their hosts' genes, which is a major indicator of HS. Our comparative genomic analysis suggests that, at least for supergroups A and B, HS is more frequent than expected, occurring even between distantly-related species.**

Wolbachia is a genus of gram-negative intracellular endosymbiotic bacteria. First isolated from *Culex pipiens*, it is currently estimated to be found in 20–66% of all insect species[1]. Moreover, it also infects species of filarial nematodes, arachnids, and terrestrial crustaceans[2]. *Wolbachia* belongs to the Rickettisiales order, the same order of vertebrate pathogens transmitted by arthropod vectors, although there is no evidence of *Wolbachia* causing disease in vertebrates[3,4]. There are a myriad of *Wolbachia* lineages that differ substantially at the genomic level, but they are all classified under the umbrella of a single species *Wolbachia pipientis*. Its strains are divided into supergroups, ranging from A to U, which are defined by phylogenetic analysis using the 16S rDNA, *ftsZ* and *wsp* markers[5]. It is estimated that these supergroups diverged around 100 million years ago, first in filarial nematodes and then infecting arthropods. The supergroups A and B have only been found in arthropods so far; the C and D supergroups are specific to filarial nematodes; and the E and F supergroups are mostly found in nematodes, but are also seen in some terrestrial arthropods. The remaining supergroups are distributed among other arthropod clades[6].

Long-term evolution of *Wolbachia* and their hosts have driven the emergence of diverse ecological relationships from mutualism to parasitism, depending on the lineage/supergroup-host pair. Parasitic *Wolbachia* lineages modulate different aspects of host physiology, such as the reproductive cycle, host behaviour and pathogen susceptibility[1,7]. Nematode-infecting *Wolbachia* usually have a mutualistic association with their hosts, whereas arthropod-infecting *Wolbachia* are more associated with commensalism or parasitism, modulating their host reproductive system through male-killing, feminization, parthenogenesis or cytoplasmic incompatibility[8]. The variety of *Wolbachia* induced phenotypes on their hosts has attracted the attention of the scientific community due to its potential role in host speciation, exploitation as a biological tool of vector-borne diseases control (e.g., dengue, malaria), and to combat filarial neglected tropical diseases[9].

Wolbachia is transmitted mainly via vertical transmission, i.e., it is passed between host generations in the female oocytes[10]. *Wolbachia* is also transmitted to other individuals and species through an alternative mechanism called host shift (HS), also referred as horizontal transfer, which is the direct transmission of *Wolbachia* cells between organisms where there is no feasible mechanism of vertical transfer.

[1]Programa de Pós-Graduação em Genética e Biologia Molecular, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, Brazil. [2]Departamento de Entomologia, Instituto Aggeu Magalhães, Fundação Oswaldo Cruz, Recife, Pernambuco, Brazil. [3]Biochemistry and Molecular Biology Department, Federal University of Santa Maria, Av. Roraima 1000, Santa Maria, RS CEP 97105.900, Brazil. ✉email: elgion@base.ufsm.br

HS can alter host fitness by adding phenotypes to the new host that allow it to interact most successfully with the environment[11]. *Wolbachia* strains that can manipulate the host reproductive biology achieve a high rate of infection in the new host, substantially enhancing *Wolbachia* spreading in the next host generation[6].

As an obligatory endosymbiont that is mainly vertically transmitted, *Wolbachia* is expected to share a long evolutionary journey with their hosts. Nevertheless, there is strong evidence of *Wolbachia* ancient and recent horizontal transfer events between phylogenetically closely and distantly related host species[12–16]. Transfection experiments of *Wolbachia* were able to show its great capability to infect cells from distantly-related hosts, reinforcing the HS potential of *Wolbachia*[7,17,18]. Other characteristics that may influence HS include the ability of *Wolbachia* to survive for months in an extracellular environment, despite being an intracellular symbiont[6], as well as genome recombination, which may influence the ability of the bacterium to adapt to new environments due to genome diversification[7].

Despite the strong evidence on *Wolbachia* HS in several arthropod hosts, it is still considered a rare phenomenon[19,20]. In this study, we leveraged a large dataset of over 1000 draft and complete *Wolbachia* genomes reconstructed by Scholz et al. performing the most extensive assessment of *Wolbachia* HS so far. Our in-depth investigation of *Wolbachia*-host gene divergence revealed several long-range *Wolbachia* HS events from supergroups A and B among arthropods, suggesting HS is more frequent than normally reported for these abundant and widespread supergroups.

Materials and methods

Data. Assembled *Wolbachia* genomes were downloaded in November 2020, from https://www.ebi.ac.uk/ena/browser/view/PRJEB35167[21]; only *Wolbachia* genomes belonging to supergroups A and B were kept for analysis. Scholz retrieved existing *Wolbachia* reference genomes from refseq[22] and genbank[23], and public shotgun sequencing samples were retrieved from the NCBI sequence read archive (sra) database from all available projects involving taxa that can host *Wolbachia*. Host genomes were downloaded from https://www.ncbi.nlm.nih.gov/genome using the host species as a query term. The complete list of hosts and *Wolbachia* assemblies can be seen in Supplementary Table 1.

Orthologue identification. The orthologous genes for both *Wolbachia* and their hosts were obtained using the BUSCO v5.1.2 docker image[24] using the 'augustus' flag. The databases used were ricketisialles_odb10 and arthropoda_odb10 for *Wolbachia* and hosts, respectively. Fifty single-copy genes (Supplementary Table 1) for each strain were extracted from both searches for supergroups A and B, and single-copy genes shared between supergroups A and B to build a single evolutionary *Wolbachia* tree. In both situations, BUSCO was not able to recover 50 single-copy orthologues between all strains, the mean of recovered genes was 48.94 genes, standard deviation of 3.83 approximately. In those cases, the maximum possible number of genes for each strain was used.

Alignment. Each one of the recovered orthologous genes were codon aligned separately using MACSE v2.05[25], using the 'alignSequences' option, then all genes were concatenated by fasta identifier (ID) using the tool catfasta2phyml (available at https://github.com/nylander/catfasta2phyml) generating one fasta file with all sequences for hosts and *Wolbachia*, respectively.

Similarity analysis and descriptive statistics. The command-line tool CIAlign[26], version 1.0.9, was used to calculate the similarity between the concatenated aligned *Wolbachia* sequences, as well as for the host's aligned sequences, using the following options: '--make_similarity_matrix_input', '--make_simmatrix_keepgaps 2'. All descriptive statistics were calculated using the 'describe' method from the Python package Pandas. The 'described' method was also used to obtain the overall descriptive statistics for the mean, minimum and maximum values of the first generated statistics. The code used is available at https://github.com/Tiago-Minuzzi/wolbachia-hs.

Phylogenetic analysis. The software IQ-Tree stable release 1.6.12[27] was used to obtain the *Wolbachia* phylogeny, with the ultrafast bootstrap parameter set to 1000 and model GTR + F + R3 chosen according to BIC; the ITOL web server[28] was used to generate the tree visualisation.

Results

Phylogenetic reconstruction and lineages. *Wolbachia* assemblies were separated into supergroups based on the phylogeny by Scholz et al. A careful assessment of the alignments revealed many identical sequences between different *Wolbachia* assemblies, thus, the fasta IDs of the identical sequences were grouped, and only a single sequence was kept as a representative. After selection of representative sequences, a reduction of 1044 to 304 sequences occurred for supergroup A and from 20 to 17 for supergroup B. Most of these highly similar genomes were characterised from different populations of some model organisms, such as species from the *Drosophila* genus.

We reconstructed the *Wolbachia* phylogeny using 50 single-copy orthologues for both supergroups A and B to evaluate if the resulting tree agrees with the original dataset from Scholz et al. and showed that it matched as expected. After reconstructing the *Wolbachia* phylogeny, we grouped sequences in 23 lineages/clades that showed divergence lower than 0.02% (Supplementary Fig. 1), followed by random selection of one sequence from each *Wolbachia* lineage to estimate and compare the similarities between lineages (Fig. 1).

Supergroup A is composed of 15 lineages, occurring in 10 different hosts species. It is important to highlight that 10 out of these 15 lineages have similarities ranging from 95 to 99.9%, occurring in eight different host

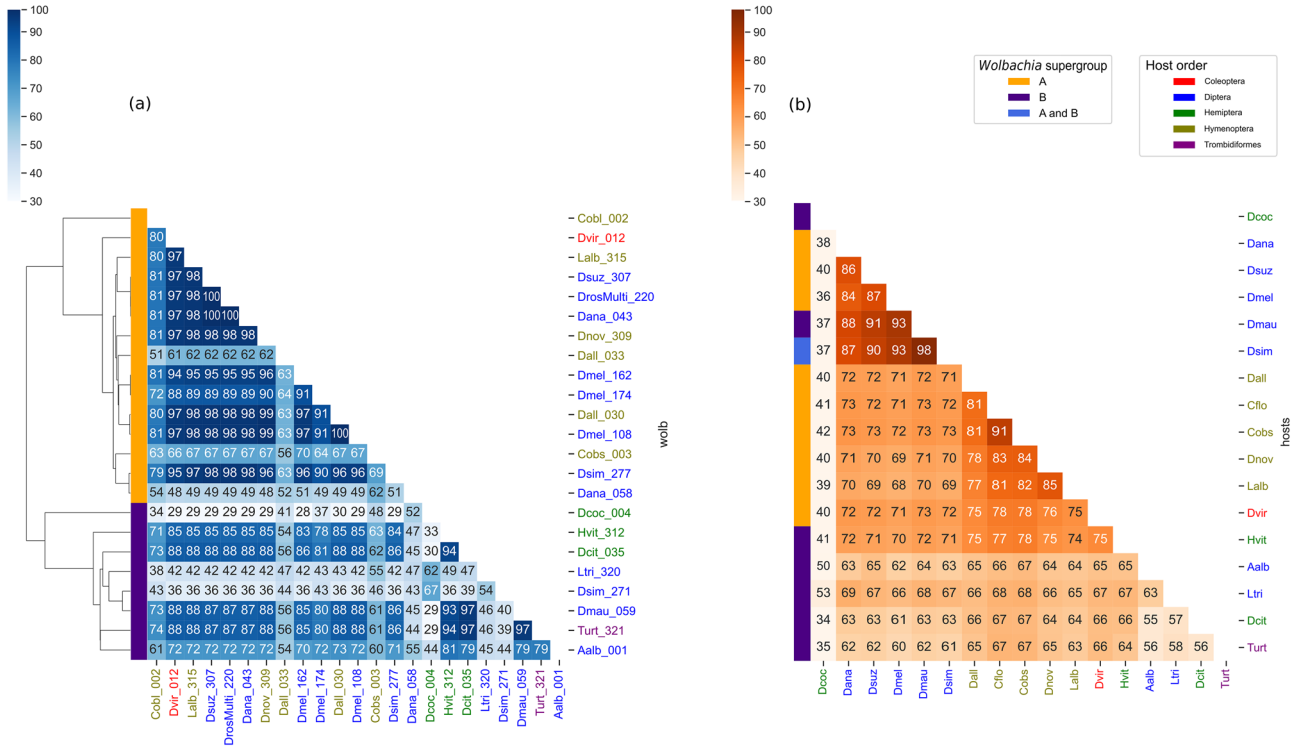


Figure 1. Heatmap showing: (**a**) *Wolbachia* similarity and (**b**) hosts similarity. *Wolbachia* heatmap shows the similarity from representatives of clades from supergroups A and B, also showing the *Wolbachia* phylogeny.

	Supergroup A				Supergroup B			
	L. albipes vs. *D. simulans*		*D. alloeum* vs. *D. melanogaster*		*T. urticae* vs. *A. albopictus*		*H. vitripennis* vs. *D. mauritiana*	
	Host	Wolb	Host	Wolb	Host	Wolb	Host	Wolb
n_genes	50	50	48	47	50	50	48	50
Mean	48.36%	98.51%	47.24%	99.87%	40.80%	94.37%	47.81%	94.17%
Std	11.77%	2.92%	12.21%	0.16%	12.07%	6.27%	10.84%	4.71%
Min	16.37%	80.48%	21.64%	99.33%	14.73%	74.35%	25.91%	73.93%
25%	40.52%	98.05%	40.45%	99.77%	32.68%	93.83%	41.66%	92.46%
50%	48.18%	99.42%	46.63%	99.99%	39.42%	96.54%	46.42%	95.10%
75%	57.29%	99.99%	54.93%	99.99%	49.57%	98.78%	55.95%	97.32%
Max	68.49%	99.99%	68.71%	99.99%	68.80%	99.67%	71.33%	99.53%

Table 1. Descriptive statistics of pairwise gene sequence similarity of *Wolbachia* and hosts. *n_genes* number of genes, *std* standard deviation, *min* minimum value found, *max* maximum value found.

species, some of them as evolutionarily distant as Hymenoptera, Coleoptera and Diptera. These groups showed lower genetic similarity, ranging from 60 to 80% when comparing host genes (Fig. 1b). Supergroup B is composed of eight lineages found in eight different hosts. Four of these species belonging to distantly related taxa such as Acari, Hemiptera and Diptera showed *Wolbachia* gene similarities ranging from 93 to 97%. The graphical representation of gene alignments for *Wolbachia* supergroup A and supergroup B, and their hosts (Supplementary Figs. 2, 3; Supplementary Tables 2, 3) shows the high similarity within each *Wolbachia* supergroup and the lower similarity of host genes.

Pairwise gene sequence similarity. Pairwise gene sequence similarity analysis (Table 1) of *Wolbachia* and host orthologues shows striking differences (Fig. 2), corroborating the concatenated divergence analysis shown in Fig. 1. For supergroup A, the mean similarity between the Hymenopteran *Lasioglossum albipes Wolbachia* (assembly WOLB0007) and dipteran *Drosophila simulans Wolbachia* (WOLB0926) orthologues was 98.51% (minimum 80.48% and maximum 99.9%); the similarity between host orthologues was 48.36% (minimum similarity 16.37% and maximum similarity 68.49%). For *Wolbachia* of Hymenopteran *Diachasma alloeum* and dipteran *Drosophila melanogaster*, WOLB1002 and WOLB0092, respectively, the mean similar-

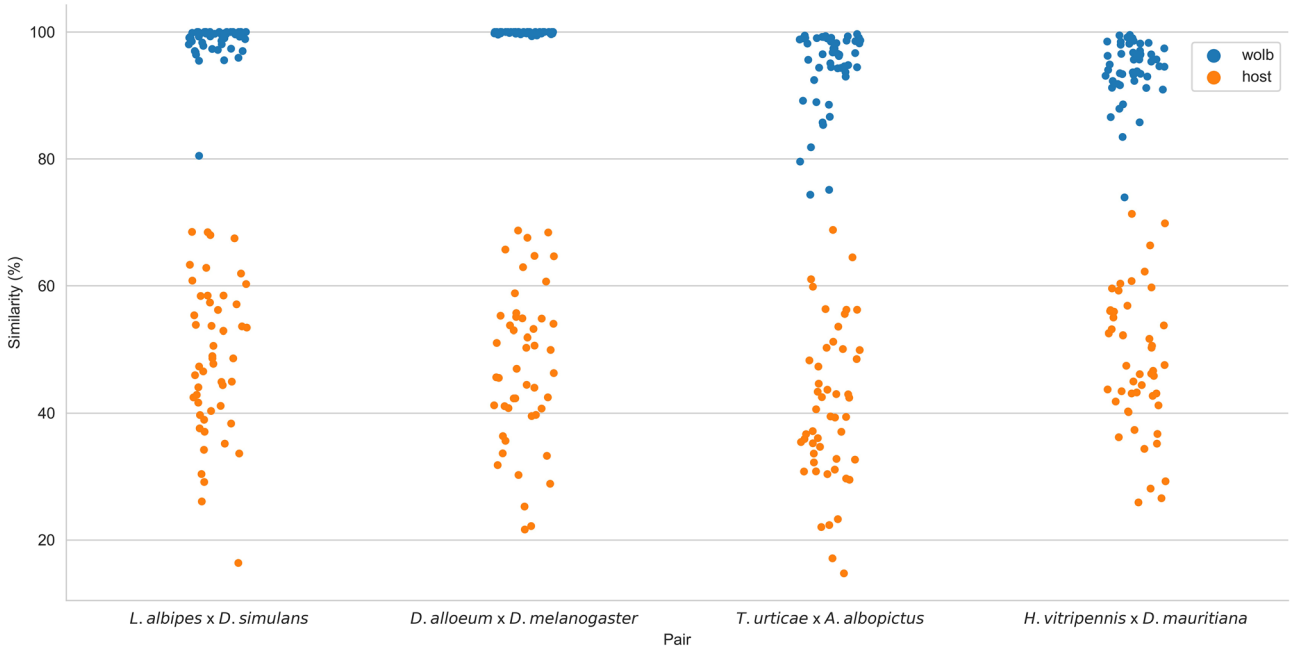


Figure 2. Pairwise gene similarity of *Wolbachia* and hosts. Each dot represents a gene pair (blue—*Wolbachia* genes; orange—host genes). It shows a higher similarity of *Wolbachia* orthologues when compared with their hosts orthologues similarity.

ity was 99.87% (minimum 99.33% and maximum similarity 99.9%); host mean similarity values were 47.24% (minimum and maximum values, 21.64% and 68.71%, respectively).

For supergroup B, the *Wolbachia* found infecting the arachnid *Tetranychus urticae* (WOLB0958) and the strain infecting the insect *Aedes albopictus Wolbachia* (WOLB1128) showed a mean orthologue similarity of 94.37% (minimum 74.35% and maximum 99.67%), while the similarity between host orthologues showed a 40.80% mean similarity (minimum 14.73% and maximum 60.80%). The mean similarity for *Wolbachia* orthologues of the hemipteran *Homalodisca vitripennis* and dipteran *Drosophila mauritiana*, assemblies WOLB0957 and WOLB0080, respectively, was 94.17% (minimum 73.93% and maximum 99.53%); the mean similarity for host orthologues was 47.81% (minimum 25.91% and maximum 71.33%). To more clearly visualise the differences between the hosts and bacteria orthologous gene divergences, they are presented as strip plots for four pairwise species comparisons (Fig. 2). Considering that *Wolbachia* mutation follow their hosts' molecular clock, as demonstrated by the correlation of *Wolbachia* and the 18S rRNA gene evolution[21], we can directly compare the evolution through time of *Wolbachia* and host genes, which demonstrates that the host genes are significantly more divergent than the *Wolbachia* genes.

Supergroup A overall similarity. From the supergroup A similarity table (Supplementary Table 2), we calculated the descriptive statistical values for *Wolbachia* similarity within the supergroup for the following examples. *Wolbachia* from *Diabrotica virgifera*, order Coleoptera, showed a mean similarity of 84.38% with the Wolbachia from *Diachasma alloeum*, order Hymenoptera, with a maximum mean of 97%, a minimum mean of 60.15%, and a mode of maximum values of 97.19% (Supplementary Table 5). In *D. virgifera* and *Drosophila melanogaster* (Diptera) *Wolbachia,* the mean similarity, in many cases, is greater than 96%, reaching maximum values greater than 97%, with an overall mean similarity of 89.08%, mode of maximum values of 97.2% in a comparison of 150 *D. melanogaster* and 22 *D. virgifera Wolbachia* (Supplementary Table 6). *D. virgifera Wolbachia* has an overall mean similarity of 96.05% with *Dufourea novaeangliae* (Hymenoptera) *Wolbachia* (Supplementary Table 7). The overall mean similarity between *D. virgifera* and *Drosophila ananassae Wolbachia*, is 90.37%, with a mean of max values of 90.9%, and a mode of max values of 96.40% (Supplementary Table 8).

Supergroup B overall similarity. In Supergroup B, orthologue similarity analysis (Supplementary Table 3) and descriptive statistics (Supplementary Table 9) show that *Wolbachia* from Hemiptera *Diaphorina citri* has a mean similarity of 93.88% with *Wolbachia* from *Tetranychus urticae*, order Trombidiformes, Class Arachnida (minimum 88.56% and maximum 95.67%). The *D.* citri and *Drosophila mauritiana Wolbachia* similarity was 93.04% (minimum 88.19% and maximum 94.7%); *D. citri Wolbachia* similarity with *Wolbachia* from *Homalodisca vitripennis* (Hemiptera) was 93.49% (minimum 88.34% and maximum 95.25%); and *D. citri Wolbachia* has a mean similarity of 90.92% with *Wolbachia* from *A. albopictus* (minimum 86.45% and maximum 93.08%).

Discussion

Wolbachia is the most widespread endosymbiotic organism in arthropods. One of the main features thought to be responsible for its successful long-term persistence in nature is its ability to manipulate host physiology and specifically host reproductive biology, conferring fitness benefits to *Wolbachia* and eventually to its host, including, for instance, increased pathogen resistance[29]. Maternal transmission, or vertical transfer, is the main process used by *Wolbachia* to infect a new host offspring, which, through evolutionary time, may allow these bacteria to prevail in different host species. Additionally, *Wolbachia* infection also can occur via hybridisation and introgression of similarly related species, or by HS between closely and distantly related species[30].

Although *Wolbachia* HS is a well-documented phenomenon[6,7,18,31–34], a large amount of the literature depicts it as a rare event[19,20]. Our comparative genomic analyses of several *Wolbachia* strains and their hosts reinforce the occurrence of HS in these bacteria, showing many cases in which different host species share *Wolbachia* more similar than would be expected by long-term coevolution of vertically transmitted endosymbionts with their hosts. However, the novel finding of our data is that HS, at least for *Wolbachia* supergroups A and B, seems to be more frequent than expected.

Six out of 17 host species bearing *Wolbachia* supergroups A and B showed *Wolbachia* similarity higher than 95%, pointing out that this *Wolbachia* was shared by HS very recently, even between phylogenetically distant host taxa as Hymenoptera, Coleoptera and Diptera (Fig. 3a). Additionally, for supergroup B, four host species as phylogenetically distant as Acari, Diptera and Hemiptera share a *Wolbachia* lineage that is more than 93% similar at the nucleotide level (Fig. 3b). Therefore, from the 17 host species analysed, at least 10 (58.8%) shared *Wolbachia* lineages by HS. Thus, we ask: is HS a rare phenomenon in *Wolbachia* evolution?

HS depends on specific environmental conditions to happen, alongside the ability of a *Wolbachia* strain to infect a new host and maintain the infection[7]. It has been hypothesised that the closer the phylogenetic relationship of the hosts, the more likely HS is to occur[34], which may induce novel phenotypes in the new host[18]. The underlying mechanisms of HS are not yet fully understood, leading it to be overlooked on many occasions.

Wolbachia migrates from somatic tissues to germline cells during the host's development, transferred by cell-to-cell contact via phagocytic/endocytic machinery. Yet, in cell culture, *Wolbachia* can infect *Wolbachia*-free cells independently of cell contact through the culture medium[31]. Infection by *Wolbachia*, which is present in the haemolymph, can occur by contact with excretions or injuries of an infected host to an uninfected host[34]; thus, shared food sources and feeding habits are plausible pathways for *Wolbachia* HS between different hosts[35]. Another factor contributing to *Wolbachia* HS is predation, where ingested larvae contaminate the uninfected host, crossing the digestive system epithelium and colonising the future ovarian stem cells[36]. Parasitoid-host interactions are well documented as another route *Wolbachia* uses to move between species[12,15,18]. Among the organisms analysed in the present study, some already showed previous evidence of HS, and are either parasitoids, e.g., *Diachasma alloeum*[11], or parasitised by a parasitoid, for example in *Drosophila melanogaster* and other *Drosophila* species[4]. HS through such interactions reinforce them as a viable mechanisms of direct *Wolbachia* transfer on a short time scale. It is important to note that, in field samples, the *Wolbachia* detected on a host may be due to sequencing reads derived from another species that are closely associated with the primary investigated host such as endoparasitoids. For instance, *Wolbachia* detected in *Ixodes ricinus*, which were actually from its endoparasitoid *Ixodiphagus hookeri*[37], and the detection of *Wolbachia* from Strepsiptera found in the Australian tephritid fruit flies[38]. Although this may occur, it should not affect the general HS pattern identified, since there is no evidence that most of the host species analysed have endoparasitoids. Also, by the amount of data analyzed in our work and the detection of high similarity between many different species as we present here, it would be very unlikely that it is the case here, thus causing any sort of analysis bias.

The phylogenetic patterns of *Wolbachia* and its hosts usually show incongruences, indicating recent HS events and successful infection of new host species[30]. We found several instances of incongruences in the phylogenetic trees of *Wolbachia* and its hosts (Supplementary Fig. 1), reinforcing the presence of HS. Moreover, our similarity analysis showed that different *Wolbachia* show high levels of similarity within the group for both supergroup A and B (Supplementary Tables 2 and 3), whilst host similarity was lower, indicating that HS is very likely to occur in natural environments, as previously suggested[32].

The order Coleoptera dates from more than 250 million years ago (mya), and the Diptera order around 200 mya[39]. In our analysis, the supergroup A of *Wolbachia* from both the Coleoptera *D. virgifera* and Diptera *D. melanogaster* showed very high similarity (Fig. 3a), considering that supergroup A dates from 76 mya[40]; HS presents itself as a strong hypothesis to explain the high similarity of *Wolbachia* from distantly related hosts. The same rationale is applied when comparing the Hemiptera (an order dating from nearly 350 mya) *D. citri* and *A. albopictus* (Diptera), in which their respective *Wolbachia* from supergroup B (dating from around 112 mya) also shows high similarity (Fig. 3b).

In the process of genome assembly of eukaryotic organisms, a common step is the removal of bacterial sequences. This process, although important for these studies, reduces the possibility of a proper assessment of symbionts HS[18], which may be related to claims of HS not being a common event. In our study, using publicly available data, we calculated the within groups similarity of *Wolbachia* from supergroups A and B, tracing a parallel with their hosts' similarity. The data showed that many *Wolbachia* from distantly related hosts share high similarity, while their hosts' core gene similarity is significantly lower, alongside a divergence between host and *Wolbachia* phylogenetic trees. We found that 58.8% of host species analysed share two particular *Wolbachia* lineages, indicating that these lineages have been acquired by HS recently and suggesting that HS events may be more frequent than previously thought. This is evidence for the HS hypothesis being a common outcome of different ecological interactions, explaining at least partially how *Wolbachia* became such a ubiquitous organism across multiple clades. In addition, epidemiological modelling of *Wolbachia* transmission demonstrated that it would not be possible to explain *Wolbachia* incidence in a broad range of clades only considering it as

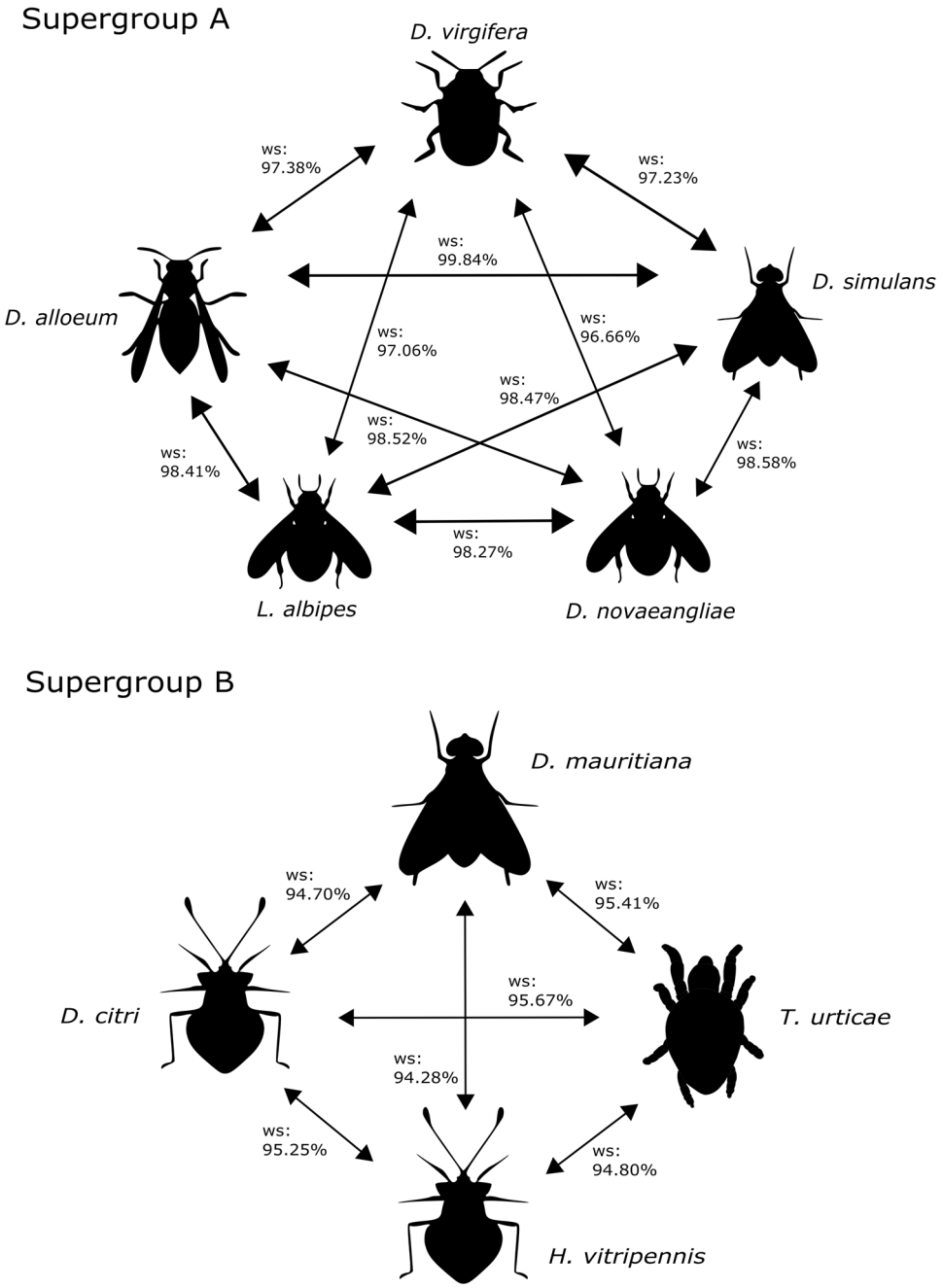


Figure 3. *Wolbachia* similarity between different hosts. The high *Wolbachia* similarity between distant related hosts is a strong evidence of HS since there is no feasible way of vertical transfer of *Wolbachia* between those hosts. ws, *Wolbachia* similarity.

vertically transmitted[41], thus it is necessary to take host shift into account to explain the spread of *Wolbachia* in phylogenetically distant hosts.

Wolbachia HS is a known event described by a wide range of literature[4,6,7,14,15,32,33], yet it is still somewhat overlooked and sometimes disbelieved as a more common mechanism[19,20,30], as it is still not very clear how it is established in some cases[13]. Nonetheless, *Wolbachia* has an arsenal of well described methods to thrive when first encountering a new host, which may explain its success jumping across clades by HS[6]. This arsenal consists of the facts that *Wolbachia* has no problem adapting to new environments[7], can, without much effort, move across cells and tissues, as it is a proficient manipulator of its hosts physiology[6,42]. Even though *Wolbachia* may cause reduced host fitness, the opposite is also true, as *Wolbachia* may alter pathogen susceptibility conferring viral protection for its hosts[43]. Also, *Wolbachia* can survive for a limited time in an extracellular environment, albeit being an obligatory intracellular endosymbiont[12,35].

By using gene similarity of over 1000 reconstructed genomes[21], alongside a phylogenetic reconstruction, we were able to bring focus to *Wolbachia* HS, estimate the event and compare it in *Wolbachia* supergroups A and B of close and distant related hosts and their *Wolbachia*, shedding more light on the importance of HS as a major player in *Wolbachia* pervasiveness on very distinctive branches of the Arthropoda tree.

Received: 12 October 2021; Accepted: 9 May 2022
Published online: 17 May 2022

References

1. Landmann, F. The *Wolbachia* endosymbionts. in *Bacteria and Intracellularity* 139–153 (American Society of Microbiology, 2019). https://doi.org/10.1128/microbiolspec.bai-0018-2019
2. Hedges, L. M., Brownlie, J. C., O'Neill, S. L. & Johnson, K. N. *Wolbachia* and virus protection in insects. *Science* **322**, 702 (2008).
3. Werren, J. H. Biology of *Wolbachia*. *Annu. Rev. Entomol.* **42**, 587–609 (1997).
4. Brown, A. N. & Lloyd, V. K. Evidence for horizontal transfer of *Wolbachia* by a *Drosophila* mite. *Exp. Appl. Acarol.* **66**, 301–311 (2015).
5. Baimai, V., Ahantarig, A. & Trinachartvanit, W. Novel supergroup U *Wolbachia* in bat mites of Thailand. *Southeast Asian J. Trop. Med. Public Health* **52**, 48–55 (2021).
6. Sanaei, E., Charlat, S. & Engelstädter, J. *Wolbachia* host shifts: Routes, mechanisms, constraints and evolutionary consequences. *Biol. Rev.* **96**, 433–453 (2021).
7. Tolley, S. J. A., Nonacs, P. & Sapountzis, P. *Wolbachia* horizontal transmission events in ants: What do we know and what can we learn?. *Front. Microbiol.* **10**, 296 (2019).
8. Lo, N., Casiraghi, M., Salati, E., Bazzocchi, C. & Bandi, C. How many *Wolbachia* supergroups exist? [2]. *Mol. Biol. Evol.* **19**, 341–346 (2002).
9. Ding, H., Yeo, H. & Puniamoorthy, N. *Wolbachia* infection in wild mosquitoes (Diptera: Culicidae): Implications for transmission modes and host-endosymbiont associations in Singapore. *Parasit. Vectors* **13**, 1–16 (2020).
10. Singh, N. D. *Wolbachia* infection associated with increased recombination in drosophila. *G3 Genes Genomes Genet.* **9**, 229–237 (2019).
11. Dhaygude, K., Nair, A., Johansson, H., Wurm, Y. & Sundström, L. The first draft genomes of the ant *Formica exsecta*, and its *Wolbachia* endosymbiont reveal extensive gene transfer from endosymbiont to host. *BMC Genomics* **20**, 1–16 (2019).
12. Le Clec'h, W. *et al.* Cannibalism and predation as paths for horizontal passage of *Wolbachia* between terrestrial isopods. *PLoS One* **8**, e60232 (2013).
13. Siozios, S., Gerth, M., Griffin, J. S. & Hurst, G. D. D. Symbiosis: *Wolbachia* host shifts in the fast lane. *Curr. Biol.* **28**, R269–R271 (2018).
14. Pimentel, A. C., Beraldo, C. S. & Cogni, R. Host-shift as the cause of emerging infectious diseases: Experimental approaches using drosophila–virus interactions. *Genet. Mol. Biol.* **44**, 1–8 (2021).
15. Duplouy, A., Pranter, R., Warren-Gash, H., Tropek, R. & Wahlberg, N. Towards unravelling *Wolbachia* global exchange: A contribution from the *Bicyclus* and *Mylothris* butterflies in the Afrotropics. *BMC Microbiol.* **20**, 319 (2020).
16. Stahlhut, J. K. *et al.* The mushroom habitat as an ecological arena for global exchange of *Wolbachia*. *Mol. Ecol.* **19**, 1940–1952 (2010).
17. Newton, I. L. G., Savytskyy, O. & Sheehan, K. B. *Wolbachia* utilize host actin for efficient maternal transmission in *Drosophila melanogaster*. *PLoS Pathog.* **11**, 1004798 (2015).
18. Ahmed, M. Z., Breinholt, J. W. & Kawahara, A. Y. Evidence for common horizontal transmission of *Wolbachia* among butterflies and moths. *BMC Evol. Biol.* **16**, 1–16 (2016).
19. Hamm, C. A. *et al.* *Wolbachia* do not live by reproductive manipulation alone: Infection polymorphism in *Drosophila suzukii* and *D Subpulchrella*. *Mol. Ecol.* **23**, 4871–4885 (2014).
20. O'Neill, S. L. *Wolbachia* mosquito control: Tested. *Science* **352**, 526 (2016).
21. Scholz, M. *et al.* Large scale genome reconstructions illuminate *Wolbachia* evolution. *Nat. Commun.* **11**, 5235–5235 (2020).
22. O'Leary, N. A. *et al.* Reference sequence (RefSeq) database at NCBI: Current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.* **44**, D733–D745 (2016).
23. Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J. & Wheeler, D. L. GenBank. *Nucleic Acids Res.* **33**, 34–38 (2005).
24. Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V. & Zdobnov, E. M. BUSCO: Assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* **31**, 3210–3212 (2015).
25. Ranwez, V., Harispe, S., Delsuc, F. & Douzery, E. J. P. MACSE: Multiple alignment of coding sequences accounting for frameshifts and stop codons. *PLoS One* **6**, 22594 (2011).
26. Tumescheit, C., Firth, A. E. & Brown, K. CIAlign—A highly customisable command line tool to clean, interpret and visualise multiple sequence alignments. *PeerJ* **10**, e12983 (2020).
27. Nguyen, L. T., Schmidt, H. A., Von Haeseler, A. & Minh, B. Q. IQ-TREE: A fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Mol. Biol. Evol.* **32**, 268–274 (2015).
28. Letunic, I. & Bork, P. Interactive Tree Of Life (iTOL) v5: An online tool for phylogenetic tree display and annotation. *Nucleic Acids Res.* **49**, W293–W296 (2021).
29. Teixeira, L., Ferreira, Á. & Ashburner, M. The bacterial symbiont *Wolbachia* induces resistance to RNA viral infections in *Drosophila melanogaster*. *PLoS Biol.* **6**, 2753–2763 (2008).
30. Turelli, M. *et al.* Rapid global spread of wRi-like *Wolbachia* across multiple *Drosophila*. *Curr. Biol.* **28**, 963-971.e8 (2018).
31. White, P. M. *et al.* Mechanisms of horizontal cell-to-cell transfer of *Wolbachia* spp. *Drosophila melanogaster*. *Appl. Environ. Microbiol.* **83**, 3425–3441 (2017).
32. Pattabhiramaiah, M., Brückner, D. & Reddy, M. Horizontal transmission of *Wolbachia* in the honeybee subspecies *Apis mellifera carnica* and its ectoparasite *Varroa destructor*. *Int. J. Environ. Sci.* **2**, 514 (2011).
33. Tseng, S. P. *et al.* Evidence for common horizontal transmission of *Wolbachia* among ants and ant crickets: Kleptoparasitism added to the list. *Microorganisms* **8**, 805 (2020).
34. Zimmermann, B. L. *et al.* Supergroup F *Wolbachia* in terrestrial isopods: Horizontal transmission from termites?. *Evol. Ecol.* **35**, 165–182 (2021).
35. Cardoso, A. & Gómez-Zurita, J. Food resource sharing of alder leaf beetle specialists (Coleoptera: Chrysomelidae) as potential insect-plant interface for horizontal transmission of endosymbionts. *Environ. Entomol.* **49**, 1402–1414 (2020).
36. Faria, V. G., Paulo, T. F. & Sucena, É. Testing cannibalism as a mechanism for horizontal transmission of *Wolbachia* in *Drosophila*. *Symbiosis* **68**, 79–85 (2016).
37. Plantard, O. *et al.* Detection of *Wolbachia* in the tick *Ixodes ricinus* is due to the presence of the hymenoptera endoparasitoid *Ixodiphagus hookeri*. *PLoS One* **7**, 1–8 (2012).

38. Towett-Kirui, S., Morrow, J. L., Close, S., Royer, J. E. & Riegler, M. Host–endoparasitoid–endosymbiont relationships: Concealed Strepsiptera provide new twist to *Wolbachia* in Australian tephritid fruit flies. *Environ. Microbiol.* **23**, 5587–5604 (2021).
39. Misof, B. *et al.* Phylogenomics resolves the timing and pattern of insect evolution. *Science* **346**, 763–767 (2014).
40. Gerth, M. & Bleidorn, C. Comparative genomics provides a timeframe for *Wolbachia* evolution and exposes a recent biotin synthesis operon transfer. *Nat. Microbiol.* **2**, 1–7 (2016).
41. Zug, R., Koehncke, A. & Hammerstein, P. Epidemiology in evolutionary time: The case of *Wolbachia* horizontal transmission between arthropod host species. *J. Evol. Biol.* **25**, 2149–2160 (2012).
42. Hague, M. T. J., Caldwell, C. N. & Cooper, B. S. Pervasive effects of *Wolbachia* on host temperature preference. *MBio* **11**, 1–15 (2020).
43. Zug, R. & Hammerstein, P. Bad guys turned nice? A critical assessment of *Wolbachia* mutualisms in arthropod hosts. *Biol. Rev. Camb. Philos. Soc.* **90**, 89–111 (2015).

Author contributions

E.L. and G.L.W. and T.M.F.F.G. designed the project, analyzed the data and wrote the manuscript. T.M.F.F.G. prepared the original artwork. All authors have made intellectual contributions to the research project and approved the final manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-12299-x.

Correspondence and requests for materials should be addressed to E.L.S.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.