UNIVERSIDAD AUTÓNOMA DE MANIZALES

VICERRECTORÍA ACADÉMICA

UNIDAD DE INVESTIGACIÓN

UNIDAD DE POSGRADOS

UNIVERSIDAD AUTÓNOMA DE MANIZALES

PROYECTO: Clasificación de LTR retrotransposones usando Redes Neuronales Convolucionales.

GRUPO DE INVESTIGACIÓN:

- Ingeniería de Software
- Automática

ESTUDIANTE: Simón Gaviria Orrego

TUTOR DE TESIS: Simón Orozco Arias

CO TUTOR DE TESIS: Reinel Tabares Soto

DATOS DE IDENTIFICACIÓN:

Email: simon.gaviriao@autonoma.edu.co

CC. 1058821551

AÑO: 2020

1. RESUMEN

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. Después de cada capa, por lo general se añade una función para realizar un mapeo causal no-lineal. Como cualquier red empleada para clasificación, al principio estas redes tienen una fase de extracción de características, compuesta de neuronas convolucionales, luego hay una reducción por muestreo y al final se tienen neuronas de perceptrón más sencillas para realizar la clasificación final sobre las características extraídas.

En el presente proyecto se implementaron y afinaron dos redes neuronales convolucionales, la primera basada en la publicada por Da Cruz et al., 2019 (TERL)), la cual fue utilizada en la clasificación de elementos transponibles de todas las clases, pero hasta el nivel de superfamilias usando convoluciones en dos dimensiones y la segunda es DeepTE, la cual clasifica TE desconocidos utilizando redes neuronales convolucionales (Haidong Yan, 2020), pero aplicando convoluciones en una sola dimensión.

Partiendo de la arquitectura TERL, se afinaron los híper-parámetros más comunes como cantidad de filtros, tamaño de cada filtro, tamaño del stride, tamaño del padding, diferentes capas de pooling, cantidad de capas convolucionales y completamente conectadas, cantidad de neuronas por capa (en las completamente conectadas), funciones de activación, algoritmos de optimización, métodos de regularización, entre otros. Esta arquitectura fue entrenada y afinada para el problema específico de clasificar LTR retrotransposones a nivel de linajes sin la necesidad de métodos basados en alineamientos, usando una base de datos creada por el equipo y que consta de más de 40 mil muestras, cada una transformada en una representación en dos dimensiones.

Finalmente se implementó la arquitectura DeepTE la cual usó una base de datos también creada por el equipo, pero esta fue representada mediante vectores que contenían la frecuencia de k-mers presentes en el elemento transponible.

Para la ejecución de estas arquitecturas, fue necesario el uso de una Workstation con 128 Gb en RAM, 32 cores y una tarjeta gráfica RTX 2080 súper. Para medir el rendimiento de la arquitectura, se usaron métricas bien conocidas en la comunidad como precisión, sensibilidad, especificidad, F1-score, entre otras.

PALABRAS CLAVES: Elementos transponibles, LTR Retrotransposones, Bioinformática, redes neuronales convolucionales.

ABSTRACT:

Convolutional neural networks consist of multiple layers of convolutional filters of one or more dimensions. After each layer, a function is usually added to perform non-linear causal mapping. Like any network used for classification, at the beginning these networks have a characteristic extraction phase, composed of convolutional neurons, then there is a reduction by sampling and at the end there are simpler perceptron neurons to perform the final classification on the extracted characteristics.

In this project, two convolutional neural networks were implemented and fine-tuned, the first based on the one published by Da Cruz et al., 2019 (TERL)), which was used in the classification of transposable elements of all classes, but up to the level of superfamilies using convolutions in two dimensions and the second is DeepTE, which classifies unknown TE using convolutional neural networks (Haidong Yan, 2020), utilizing convolutions in one dimension.

Starting from the TERL architecture, the most common hyper-parameters such as number of filters, size of each filter, stride size, padding size, different pooling layers, number of convolutional and fully connected layers, number of neurons per layer were refined. (in fully connected ones), activation functions, optimization algorithms, regularization methods, among others. This architecture was trained and tuned for the specific problem of classifying LTR retrotransposons at the lineage level without the need for alignment-based methods, using a database created by the team and consisting of more than 40 thousand samples, each transformed into a two-dimensional representation.

Finally, the DeepTE architecture was implemented, which used a database also created by the team, but this was represented by vectors containing the frequency of k-mers present in the transposable element..

KEY WORDS: Transposable elements, LTR Retrotransposons, Bioinformatics, convolutional neural networks.

TABLA DE CONTENIDO

2. ÁREA PROBLEMÁTICA Y JUSTIFICACIÓN

Los elementos transponibles, también conocidos como “genes saltarines” son secuencias de ADN capaces de cambiar de posición en el genoma por sí mismos. La científica estadounidense Barbara McClintock propuso su existencia en sus estudios en maíz a mediados del siglo XX y, desde entonces, todavía se está debatiendo su impacto en la evolución de muchos organismos eucariotas. Esto es porque, precisamente por su capacidad de “saltar” a otras zonas del genoma, los transposones son capaces de alterar de muchas formas la expresión de los genes o generar mutaciones y, por tanto, son potenciales elementos evolutivos. (Megía Gonazles, 2020)

La agrupación de elementos por sus características estructurales comunes y el mecanismo de transposición funcionan bien para grupos que tienen estructuras uniformes y mecanismos de integración bien definidos (Eickbush y Jamburuthugoda, 2008), pero hay pocas características compartidas a la hora de analizar otros tipos.

Actualmente se encuentra publicada una arquitectura basada en redes neuronales convolucionales para la clasificación de elementos transponibles a nivel de súper-familias (TERL). Sin embargo, el método propuesto no llega a niveles más profundos de clasificación y no se especializa en plantas, sino que usa información de diferentes organismos (Cruz, M. H. P, 2020). Por otro lado, los autores demostraron que las redes neuronales convolucionales profundas superaran todos los métodos bioinformáticos, especialmente en tiempos de ejecución. Esto se debe a la aceleración de la GPU, que proporcionan marcos como Tensorflow (Cruz, M. H. P, 2020). La Arquitectura TERL utiliza datos representados matricialmente en cuales se tienen las bases y su posición en el ADN de los elementos transponibles.

Otra arquitectura encontrada en la literatura es DeepTE, la cual clasifica TE desconocidos utilizando redes neuronales convolucionales. DeepTE usa secuencias transferidas a vectores de entrada basados en recuentos de k-mer. Utiliza un proceso de clasificación estructurado en árbol donde ocho modelos son entrenados para clasificar las TE en superfamilias y órdenes (Haidong Yan , 2020).

Dada la complejidad del proceso, toma sentido diseñar y generar un modelo capaz de realizar esta tarea de forma automática, disminuyendo el tiempo de trabajo y las actividades manuales que en la actualidad son realizadas por especialistas en distintas áreas. La creación de herramientas de este tipo es de vital importancia en la actualidad debido a la gran cantidad de proyectos de secuenciación masiva que se están desarrollando y a la enorme cantidad de genomas de plantas que se liberan en bases de datos cada año.

3. REFERENTE TEÓRICO

Los elementos transponibles (TEs) fueron descubiertos por primera vez por Barbara McClintock mientras experimentaba con maíz en 1944 (Makałowski et al., 2012). Es bien conocido que estos elementos cubren una gran parte de los genomas de los eucariotas y juegan un papel importante dentro de ellos (Dashti y Masoudi-Nejad, 2010). Los retrotransposones son la clase de elemento repetitivo más abundante ya que proliferan a través de un mecanismo de copiar y pegar mediado por el ARN, aumentando rápidamente su número de copias (Q. Li et al., 2018; Schulman, 2013). Por ejemplo, en el maíz y la caña de azúcar, los retrotransposones representan aproximadamente el 40-75% de la información genómica (Negi et al., 2016). Esto es especialmente cierto en el caso de los LTR retrotransposones. Los TEs son conocidos por su variabilidad en las estructuras, mecanismos de replicación, funciones y ubicaciones dentro de los genomas.

Los retrotransposones o Clase I se dividen en cuatro órdenes según la clasificación de Wicker (Wicker et al., 2007):

- LTR retrotransposones
- no-LTR retrotransposones
- PLEs
- DIRS

Todos ellos tienen diferencias significativas en su estructura, la presencia y organización de dominios, motifs o regiones reguladoras enzimáticas y en su ciclo de vida (Kejnovsky et al., 2015).

En los últimos años se ha realizado un esfuerzo considerable para crear un sistema unificado de clasificación. Uno de los métodos más aceptados fue el sistema de clasificación jerárquica que subdividió a los TEs en clases, subclases, órdenes, súper-familias, linajes y familias propuesto por Wicker y Keller en 2007 (Paz et al., 2017; Wicker et al., 2007). Siguiendo la propuesta de Wicker, los TEs pueden clasificarse en clases según su mecanismo de replicación, y pueden dividirse en retrotransposones (Clase 1) y transposones de ADN (Clase 2) (Huang et al., 2012). Los retrotransposones usan un mecanismo de "copiar y pegar" para replicar usando un intermediario de ARN.

Los retrotransposones son elementos genéticos que se pueden amplificar a sí mismos en un genoma y son ubicuos componentes del ADN de muchos organismos eucariotas. Según los análisis filogenéticos los retrotransposones se originaron de los intrones del grupo II procariotas después del momento que darían origen a los eucariotas. (Phillip Sanmiguel, 2020).

Ellos son una subclase de TE, y son particularmente abundantes en las plantas, donde a menudo son un componente principal de ADN nuclear. Hay dos tipos principales de retrotransposón, LTR y no LTR. Los retrotransposones se clasifican según la secuencia y el método de transposición. (Phillip Sanmiguel, 2020).

Las redes neuronales convolucionales son un algoritmo de aprendizaje profundo que puede ser aplicado a varias tareas, como la clasificación de imágenes, reconocimiento de voz y clasificación de texto. También es una representación de un algoritmo de aprendizaje, que puede aprender a representar los datos para realizar alguna tarea extrayendo características de ella. (Loncomilla, P. (2016).)

La arquitectura de la CNN se inspira en la corteza visual de los vertebrados, que es estructurado de manera que las células simples pueden reconocer estructuras sencillas, como bordes y líneas en las imágenes, y basado en lo que fue reconocido por estas simples células, otras células pueden reconocer estructuras más intrincadas (Cruz, M. H. P, 2020).

La arquitectura está compuesta básicamente por una red neural artificial con inter-unidades conectadas dispuestas de forma específica en múltiples capas, que normalmente son: entrada,

convolución, pooling y capas totalmente conectadas. La capa de entrada es la representación de los datos en bruto. La capa de convolución está compuesta por unidades de neuronas artificiales conectadas a filtros de ventana (es decir, mapas de características) que aplican la convolución a pequeñas regiones de su entrada de datos. Estas regiones se definen en función de las dimensiones del filtro, es decir, a la cantidad de datos que deben saltarse antes de la siguiente operación de convolución). Cabe anotar que los valores de las dimensiones de los filtros son aprendidos durante el proceso de entrenamiento del modelo (Bagnato, 2020). TERL es una arquitectura que permite una clasificación automática de las secuencias de TE, que se compone de cuatro pasos de preprocesamiento, una transformación de datos y redes neuronales convolucionales (Da Cruz et al., 2019).

DeepTE es una herramienta para clasificar TE basada en redes neuronales convolucionales, pero usando convoluciones en una dimensión. DeepTE contiene ocho modelos para diferentes propósitos de clasificación y también incluye una función para corregir la clasificación falsa basada en la estructura del dominio. Esta herramienta clasificó a los ET en 15-24 superfamilias de acuerdo con las secuencias de Plantas, Metazoos y Hongos. Para secuencias desconocidas, DeepTE podría distinguir no TE y TE en especies de plantas (Haidong Yan , 2020).

.

4. LOS OBJETIVOS

- **Objetivo General:** Diseñar e implementar una arquitectura de redes neuronales convolucionales para la clasificación de LTR retrotransposones en plantas.
- **Objetivos Específicos:**
  a. Definir una arquitectura de red neuronal convolucional y sus hiper parámetros.
  b. Comparar el rendimiento de la arquitectura propuesta y arquitecturas disponibles en la literatura.

5. METODOLOGÍA

Con el fin de cumplir los objetivos de este proyecto, primero fue necesaria una revisión detallada en la literatura para entender el funcionamiento de una arquitectura de redes neuronales convolucionales usada para la clasificación de LTR retrotransposones llamada TERL. (Cruz, M. H. P, 2020) y otra llamada DeepTE (Haidong Yan , 2020).

Una vez entendido el funcionamiento de dichas arquitecturas, fue necesario implementarla, en este caso se hizo sobre lenguaje Python versión 3 a través de las librerías Tensorflow 2.2 y Keras para tener una mayor facilidad. Luego de implementar la arquitectura TERL se midieron sus rendimientos bases haciendo uso de un subconjunto de datos con el fin de hacer este proceso inicial más ágil. Este subconjunto estaba conformado por los LTR retrotransposones presentes en la base de datos Repbase. A continuación, se formularon los híper parámetros que serán afinados, su orden, rango de valores, entre otros.

El conjunto de datos que se usó consta de 60 mil LTR retrotransposones clasificados en 13 linajes, encontrados en más de 195 especies de plantas. Se usó la misma transformación en dos dimensiones que la utilizada por TERL (Cruz, M. H. P, 2020). La partición de los datos fue la siguiente: 80% para entrenamiento, 10% para validación y 10% para test.

El proceso de afinamiento de dichos híper parámetros consistió en probar los diferentes rangos establecidos y en aplicar procesos de regularización tanto en capas convolucionales como en las completamente conectadas. Este proceso se hizo de forma iterativa y sistemática con el fin de hallar cuales valores de los híper parámetros mejoraban el rendimiento general de la red convolucional.

Finalmente se implementó la arquitectura DeepTE, la cual también fue implementada sobre lenguaje Python versión 3 a través de las librerías Tensorflow 2.2 y Keras.

Debido a que dicho proceso fue empírico, se siguieron recomendaciones de expertos teniendo en cuenta los valores de error de las particiones del conjunto de datos (entrenamiento, validación y test) con el fin de tomar decisiones más certeras en dicho proceso y dado el tamaño de los datos (aproximadamente 40 Gb) fue necesario el uso de una Workstation con 128 Gb en RAM, 32 cores y

una tarjeta gráfica RTX 2080 súper. Para medir el rendimiento de la arquitectura, se usaron métricas bien conocidas en la comunidad como precisión, sensibilidad, especificidad, F1-score, entre otras.

6. RESULTADOS

Cuando el modelo se sobre-entrena, este cae en el *overfitting*, el algoritmo estará considerando como válidos sólo los datos idénticos a los de que se tienen en el conjunto de entrenamiento, siendo incapaz de distinguir entradas buenas como fiables si se salen un poco de los rangos ya preestablecidos.

Cuando el modelo está afectado por *overfitting*, se ajusta muy bien a los datos de entrenamiento, pero no es capaz de inferir correctamente los datos de validación, por lo que el error de testeo es significativamente mayor que el error de entrenamiento. Todo esto se puede evidenciar el analizar la gráfica de perdida vs época que se presenta en la figura 1.

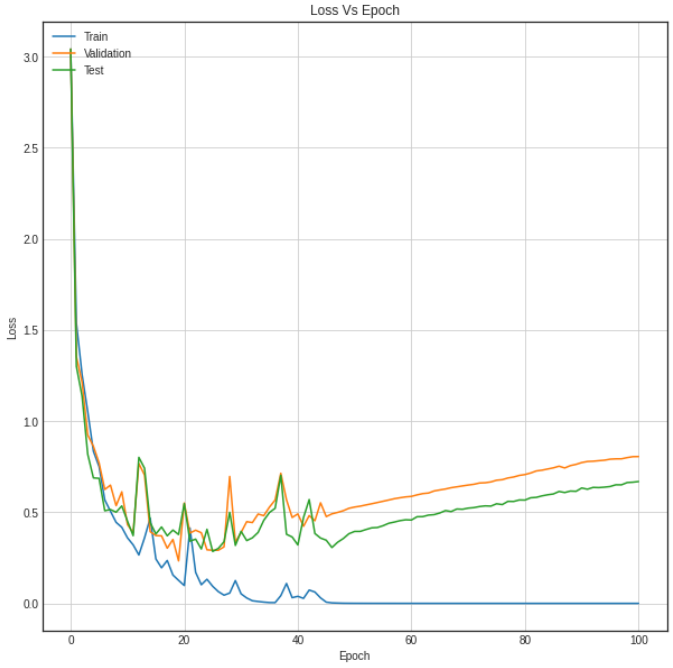


Figura 1. Pérdida obtenida en cada época usando la arquitectura TERL y entrado con Repbase.

Una manera de solucionar este problema es aplicar técnicas de regularización, cada técnica tiene diferentes impactos según la capa donde se aplique. Al analizar la arquitectura utilizada en este proyecto para la clasificación de los LTR retrotransposones a nivel de linajes se puede ver que está formada por no solo capas *fully connected* sino también capas convolucionales como se aprecia en la figura 2.

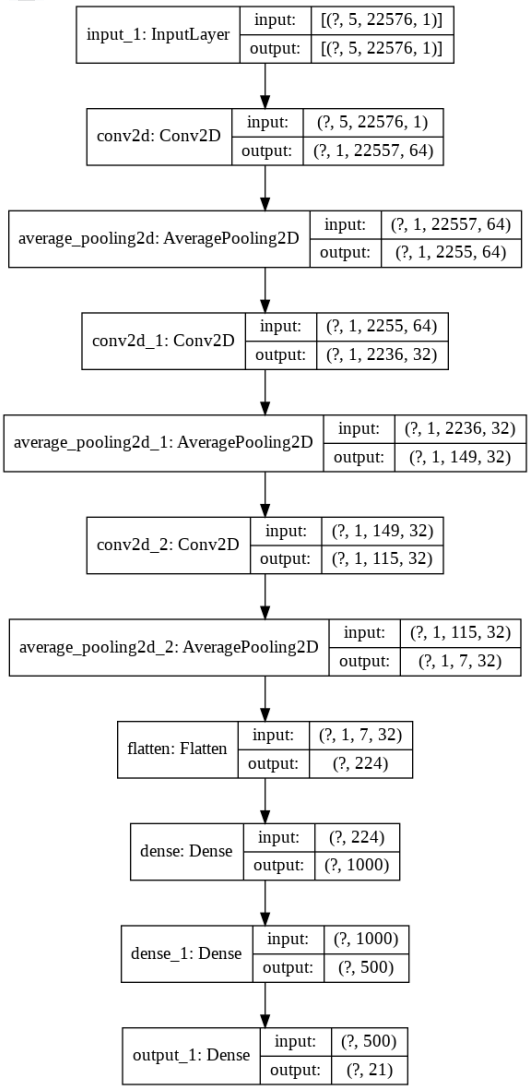


Figura 2. Arquitectura TERL inicial usada en el proyecto.

Teniendo en cuenta esta arquitectura se concluyó que se pueden utilizar los siguientes tipos de regularización:

- Normalización por lotes (Batch normalization)
- L1
- L2
- Dropout

Aplicar estas regularizaciones en las capas *fully connected*, o combinaciones de estas regularizaciones solo tuvo efectos negativos en el rendimiento de la arquitectura tanto en su rendimiento básico como de validación y test, oscilando el valor de F1-Score entre 6.34% y 14.79%.

Posteriormente se decidió aplicar estas técnicas a las capas convolucionales debido a que eran las principales encargadas de la extracción de características de los datos debido a su representación. La técnica de regularización Dropout no pudo ser aplicada a estas capas dado su funcionamiento, sin embargo, L1, L2 y normalización por lotes si fue posible. Aplicar regularización L1 y L2, o bien combinar ambas generaba errores superiores al 15% por lo tanto se descarta su uso para el mejoramiento de la arquitectura.

Finalmente se utilizó normalización por lotes en todas las capas convolucionales y se probaron diferentes parámetros hasta obtener una mejora en los rendimientos de la arquitectura y finalmente se encontró que los mejores híper parámetros en las pruebas realizadas fueron los siguientes dado que se vio una reducción en su error de validación del 6.34% hasta un 4.93%:

Tabla 1. Valor de los hiper parámetros usados en la regularización mencionada

Param Name	Value
momentum	0.2
epsilon	0.001
center	True
scale	False
trainable	True
fused	None
renorm	False

renorm_clipping	None
renorm_momentum	0.4
adjustmen	None

```
tf.keras.layers.BatchNormalization(momentum=0.2, epsilon=0.001, center=True, scale=False, trainable=True, fused=None, renorm=False, renorm_clipping=None, renorm_momentum=0.4, adjustment=None)(layers)
```

Como evidencia de esto, se presentan a continuación las gráficas de precisión, precisión vs época y perdida vs época obtenidas luego de realizar los procesos mencionados:

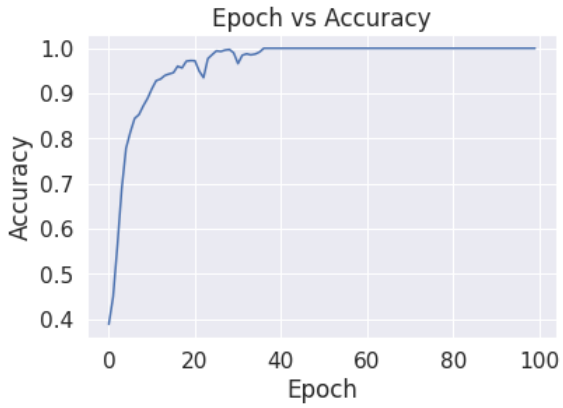


Figura 3. Precisión obtenida en cada época usando el data set de entrenamiento y la arquitectura TERL.

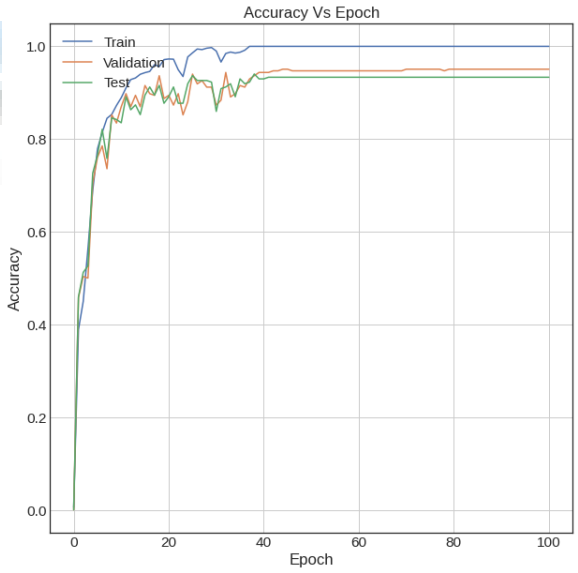


Figura 4. Precisión obtenida en cada época usando el data set de entrenamiento, validación y test y la arquitectura TERL.

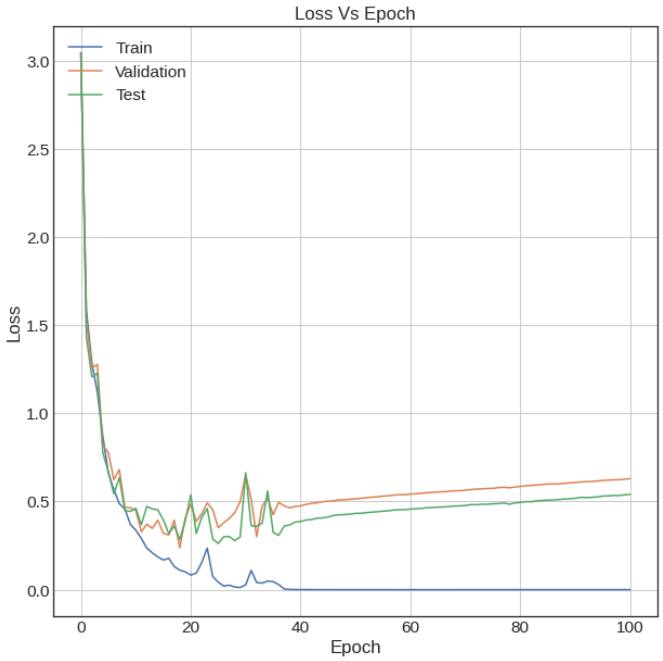
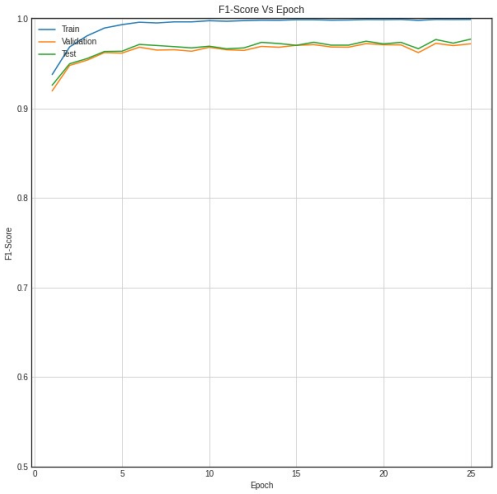


Figura 5. Pérdida obtenida en cada época usando los conjuntos de entrenamiento, validación y test

Posteriormente se realizó una prueba sobre la arquitectura DeepTE que es una arquitectura ya existente en la literatura, una de sus principales diferencias es que la representación de datos usada en ellas es 1D. Una vez ejecutada esta arquitectura con el dataset construido en el equipo se encuentra que su rendimiento es superior al obtenido por TERL, teniendo un error de validacion de solo 2.80% y un error de test de solo 2.43%. Siendo su presición de 99% como se evidencia en las figura 6 y 7.



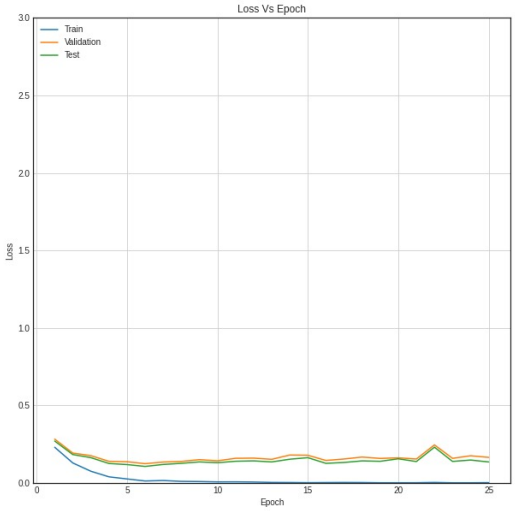


Figura 6. Precisión obtenida por cada época.

Figura 7. Pérdida obtenida por cada época.

Finalmente, las métricas obtenidas para la arquitectura TERL con las mejoras realizadas para cada uno de los linajes clasificados fueron las mostradas en la tabla 2:

Tabla 2. Métricas por linaje Arquitectura TERL afinada.

Lineage	precision	recall	f1-score	support
ALE/RETROFIT	0.97	0.99	0.98	79
ANGELA	1	0.67	0.8	3
BIANCA	0.5	1	0.67	1
IVANA/ORYCO	0.93	0.97	0.95	39
TORK	1	0.84	0.91	37

SIRE	1	0.67	0.8	6
CRM	0.91	0.91	0.91	11
GALADRIEL	1	0.6	0.75	5
REINA	0.95	0.98	0.97	60
TEKAY/DEL	0.78	1	0.88	14
ATHILA	0.8	1	0.89	8
TAT	0.95	0.91	0.93	22

Las métricas obtenidas para la arquitectura DeepTE para cada uno de los linajes clasificados fueron los mostrados en la tabla 3:

Tabla 3. Métricas por linaje Arquitectura DeepTE

Lineage	precision	recall	f1-score	support
ALE/RETROFIT	0.97	0.99	0.98	1220
ANGELA	0.96	0.94	0.95	145
BIANCA	1	0.95	0.98	166
IKEROS	1	0.43	0.6	7
IVANA/ORYCO	0.96	0.93	0.95	319
TORK	0.94	0.94	0.94	575
SIRE	0.99	0.97	0.98	325
CRM	0.97	0.92	0.94	201
GALADRIEL	1	0.74	0.85	58
REINA	0.98	0.99	0.98	497
TEKAY/DEL	0.98	0.98	0.98	1059
ATHILA	0.97	0.99	0.98	372
TAT	0.99	1	0.99	1787

7. DISCUSIÓN DE RESULTADOS

Las redes neuronales convolucionales (CNN) son un tipo de red neuronal artificial con aprendizaje supervisado que procesa sus capas imitando la corteza cerebral para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos. Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal.

El código genético son las instrucciones que le dicen a la célula cómo hacer una proteína específica. A, T, C y G, son las "letras" del código del ADN; representan los compuestos químicos adenina (A), timina (T), citosina (C) y guanina (G), respectivamente, que constituyen las bases de nucleótidos del ADN. El código para cada gen combina los cuatro compuestos químicos de diferentes maneras para formar "palabras" de tres letras las cuales especifican qué aminoácidos se necesitan en cada paso de la síntesis de una proteína (C. Brody, 2020).

Para aprovechar la arquitectura de las redes neuronales convolucionales se plantean dos tipos de representación para los datos:

- 2D
- 1D

En la representación 2D se muestran los datos de manera matricial, las columnas de la matriz representan las bases de los elementos transponibles y en las filas se ubican los posibles valores: A, T, C, G o N, donde N es un nucleótido desconocido que puede estar en alguna de las posiciones posibles.

En las posiciones que corresponde un compuesto químico se asigna 1, de lo contrario 0. Para comprender mejor esta representación se presenta la tabla 2:

Tabla 4. Representación 2D de los datos

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

La representación 2D es conveniente debido a que se tendrá siempre clara la posición de los compuestos, aunque esto implica que el procesamiento de los datos tomará más tiempo. Es importante tener en cuenta que la mayoría de los datos son 0, por lo tanto, no aportan al modelo.

Aunque estos datos en valor 0 no aportan al modelo, su existencia hace que aumente considerablemente el tamaño de los datos, y esto genera limitantes a la hora de usar la GPU y la RAM.

Para la representación 1D se tienen frecuencias o conteos de k-mers, los cuales son secuencias o cadenas de k nucleótidos que contiene el elemento transponible, la representación mencionada se puede visualizar de la siguiente manera:

Tabla 5. Representación 1D de los datos

A	C	T	G	AA	AC	AT	AG	CC	CA	...
50	40	80	33	16	18	20	22	17	18	

La representación 1D tiene como ventaja que el procesamiento de los datos es más rápido, sin embargo se desconocen las posiciones de los nucleotidos.

En la arquitectura TERL se utiliza la representación 2D de los datos mientras que en la arquitectura DeepTE se utliiza la representación 1D. La arquitectura 1D resulta ser más precisa dado que el enfoque k-mer se puede aplicar directamente a secuencias con diferentes tamaños y no es necesario corregir su ubicación dentro de una sencuencia.

La arquitectura TERL, usando el conjunto de datos creado por el equipo, tuvo un error de validación y test de 6.34% y 6.67% respectivamente, siendo su error de validación mayor a la alcanzada por la nueva arquitectura luego de realizar un afinamiento de sus híper-parametros y aplicar diversas técnicas de regularización, cuyo error de validación fue de 4.93%.

Sin embargo, ninguna de las dos arquitecturas anteriormente mencionadas tuvo el rendimiento de la arquitectura DeepTE estudiada, que además de su mejora en el rendimiento debido al cambio de la representación de los datos, posee un error de validación y test de 2.80% y 2.43% respectivamente y un valor de F1-Score del 99%.

8. CONCLUSIONES

Luego de revisar en la literatura se encontró que la arquitectura TERL clasifica de manera aceptable los elementos transponibles. Debido a esto, se implementó y analizó su rendimiento, sin embargo,

este pudo ser mejorado realizando un afinamiento en sus híper parámetros más relevantes obteniendo así una reducción en su error de conjunto de validación aproximadamente del 2%.

Esta arquitectura fue comparada con la arquitectura DeepTE, la cual hace uso de una representación 1D de los elementos transponibles analizados. Esta arquitectura obtuvo un mejor rendimiento en todos los aspectos, esto se debe a la naturalidad en la representación de los datos haciendo uso de los k-mers. Además, no es tan exigente computacionalmente y no implica tanto uso de recursos debido que no se tienen que procesar la cantidad de datos en 0 que se encontraban presentes en la representación 2D y que no daban ningún valor al modelo, esto, además, permite que las base de datos utilizadas sean menos pesadas.

9. RECOMENDACIONES

Para realizar avances de manera más ágil en la clasificación de LTR Retrotransposones se recomienda hacer uso del lenguaje de programación Python 3 debido a su gran abundancia de bibliotecas y marcos que facilitan la codificación y ahorran tiempo de desarrollo. Además de esto, gracias a su facilidad por permitir escribir código legible, garantiza un ágil desarrollo de los procesos y facilita el trabajo colaborativo gracias a su sencilla sintaxis.

También se recomienda el uso de la biblioteca Keras, dado que proporciona bloques modulares sobre los que se pueden desarrollar modelos complejos de aprendizaje profundo. A diferencia de los frameworks, este software de código abierto no se utiliza para operaciones sencillas de bajo nivel, sino que utiliza las bibliotecas de los frameworks de aprendizaje automático vinculadas (como tensorflow), que en cierto modo actúan como un motor de backend para Keras.

Dado que los procesos de machine learning y Deep learning suelen ser demasiado exigentes, a veces es difícil contar con máquinas lo suficientemente potentes para soportar las tareas a realizar, hacer uso de la herramienta Google Colaboratory garantiza tener a disposición un entorno de jupyter notebook que además de no requerir configuración es completamente gratuito. Este entorno permite el uso de una GPU y además proporciona una capacidad de memoria RAM suficiente para realizar pruebas.

10. EVIDENCIA DE RESULTADOS EN GENERACIÓN DE CONOCIMIENTO, FORTALECIMIENTO DE LA CAPACIDAD CIENTÍFICA Y APROPIACIÓN SOCIAL DEL CONOCIMIENTO, FORMACIÓN (VER ANEXO 1)

Relacionados con la generación de conocimiento y/o nuevos desarrollos tecnológicos:

Resultado/Producto esperado	Indicador	Beneficiario
Documento de diseño	1	Comunidad Cientifica y Comunidad Educativa UAM
Implementación	1	Comunidad Cientifica y Comunidad Educativa UAM

11. IMPACTOS LOGRADOS

Impacto esperado	Plazo (años) después de finalizado el proyecto: corto (1-4), mediano (5-9), largo (10 o más)	Indicador verificable	Supuestos[1]
los resultados esperados contribuirán con información relevante sobre el	Corto	Número de investigaciones que	Se seguirán secuenciando plantas

[1] Los supuestos indican los acontecimientos, las condiciones o las decisiones, necesarios para que se logre el impacto esperado.

aprovechamiento de técnicas de redes neuronales convolucionales para la clasificación de LTR retrotransposones		toman como referente la arquitectura planteada	al ritmo actual o incluso mayor
Estas herramientas aportarán al entendimiento de los genomas de las plantas que se están secuenciando actualmente, beneficiando no solamente a los investigadores interesados en las dinámicas e impactos de los TEs, sino también a los estudios de los genes y sus actividades.	mediano	Número de investigaciones que toman como referente la arquitectura planteada	Se seguirán estudiando las actividades de los genes
industrias agroindustriales podrán analizar de forma automática y más ágil cuales mutaciones son las causantes de un fenotipo de interés para obtener beneficios económicos y acelerar los procesos de selección de variedades	Mediano	Número de industrias que usan la presente investigación para analizar mutaciones causantes de un fenotipo de interés	Las industrias agroindustriales seguirán buscando mutaciones en fenotipos con el fin de obtener mayores beneficios económicos

12. BIBLIOGRAFÍA

Bousios A., Minga E., Kalitsou N., Pantermali M., Tsaballa A., Darzentas N. (2012). MASiVEdb: the Sirevirus Plant Retrotransposon Database. BMC Genomics, 13(1): 158. https://doi.org/10.1186/1471-2164-13-158

Chen L., Zhang Y.-H., Huang G., Pan X., Wang S., Huang T., Cai Y.-D. (2018). Discriminating cirRNAs from other lncRNAs using a hierarchical extreme learning machine (H-ELM) algorithm with feature selection. Mol. Genet. Genomics, 293(1): 137–149. https://doi.org/10.1007/s00438-017-1372-7

Cruz, M. H. P. (2020, 1 enero). TERL: Classification of Transposable Elements by Convolutional Neural Networks. bioRxiv. https://www.biorxiv.org/content/10.1101/2020.03.25.000935v1.abstract

Dashti T. H., Masoudi-Nejad A. (2010). Mining Biological Repetitive Sequences Using Support Vector Machines and Fuzzy SVM. Iran. J. Chem. Chem. Eng. ENGLISH Ed., 29(4): 1–17.

Eickbush T. H., Jamburuthugoda V. K. (2008). The diversity of retrotransposons and the properties of their reverse transcriptases. VIRUS Res., 134(1–2): 221–234. https://doi.org/10.1016/j.virusres.2007.12.010

Garbus I., Romero J. R., Valarik M., Vanžurová H., Karafiátová M., Cáccamo M., … Echenique V. (2015). Characterization of repetitive DNA landscape in wheat homeologous group 4 chromosomes. BMC Genomics, 16(1): 375. https://doi.org/10.1186/s12864-015-1579-0

Hermann D., Egue F., Tastard E., Nguyen D.-H., Casse N., Caruso A., … Rouault J. D. (2014). An introduction to the vast world of transposable elements - what about the diatoms? DIATOM Res., 29(1): 91–104. https://doi.org/10.1080/0269249X.2013.877083

Jiang N. (2013). Overview of Repeat Annotation and De Novo Repeat Identification. En: pp. 275–287. https://doi.org/10.1007/978-1-62703-568-2_20

Jiang S.-Y., Ramachandran S. (2013). Genome-wide survey and comparative analysis of LTR retrotransposons and their captured genes in rice and sorghum. PLoS One, 8(7): e71118. https://doi.org/10.1371/journal.pone.0071118

Kejnovsky E., Tokan V., Lexa M. (2015). Transposable elements and G-quadruplexes. Chromosome Res., 23(3): 615–623. https://doi.org/10.1007/s10577-015-9491-7

Makałowski W., Pande A., Gotea V., Makałowska I. (2012). Transposable Elements and Their identification. En: pp. 337–359. https://doi.org/10.1007/978-1-61779-582-4_12

Makarevitch I., Waters A. J., West P. T., Stitzer M., Hirsch C. N., Ross-Ibarra J., Springer N. M. (2015). Transposable Elements Contribute to Activation of Maize Genes in Response to Abiotic Stress. PLoS Genet., 11(1). https://doi.org/10.1371/journal.pgen.1004915

Negi P., Rai A. N., Suprasanna P. (2016). Moving through the Stressed Genome: Emerging Regulatory Roles for Transposons in Plant Stress Response. Front. Plant Sci., 7. https://doi.org/10.3389/fpls.2016.01448

Rawal K., Ramaswamy R. (2011). Genome-wide analysis of mobile genetic element insertion sites. Nucleic Acids Res., 39(16): 6864–6878. https://doi.org/10.1093/nar/gkr337

Schulman A. H. (2012). Hitching a Ride: Nonautonomous Retrotransposons and Parasitism as a Lifestyle. En: pp. 71–88. https://doi.org/10.1007/978-3-642-31842-9_5

Schulman A. H. (2013). Retrotransposon replication in plants. Curr. Opin. Virol., 3(6): 604–614. https://doi.org/10.1016/j.coviro.2013.08.009

Valencia J. D., Girgis H. Z. (2019). LtrDetector: A tool-suite for detecting long terminal repeat retrotransposons de-novo. BMC Genomics, 20(1): 450.

Wicker, T., Sabot, F., Hua-Van, A. et al. A unified classification system for eukaryotic transposable elements. Nat Rev Genet 8, 973–982 (2007). https://doi.org/10.1038/nrg2165

Genome.gov. 2020. Código Genético | NHGRI. [online] Available at: <https://www.genome.gov/es/genetics-glossary/Codigo-genetico#:~:text=A%2C%20T%2C%20C%20y%20G,bases%20de%20nucleótidos%20del%20ADN.> [Accessed 18 November 2020].

Biorxiv.org. 2020. [online] Available at: <https://www.biorxiv.org/content/10.1101/2020.01.27.921874v1.full.pdf> [Accessed 18 November 2020].

Genotipia.com. 2020. Elementos Transponibles: Los “Saltimbanquis” Del Genoma -. [online] Available at: <https://genotipia.com/elementos-transponibles/#:~:text=¿Qué%20son%20los%20elementos%20transponibles,el%20genoma%20por%20sí%20mismos.> [Accessed 19 November 2020].

Biorxiv.org. 2020. Deepte: A Computational Method For De Novo Classification Of Transposons With Convolutional Neural Network. [online] Available at: <https://www.biorxiv.org/content/10.1101/2020.01.27.921874v1.full.pdf> [Accessed 19 November 2020].

2020. Evidence That A Recent Increase In Maize Genome Size Was Caused By The Massive Amplification Of Intergene Retrotransposons. [online] Available at: <https://watermark.silverchair.com/37.pdf> [Accessed 19 November 2020].

Loncomilla, P. (2016). Deep learning: Redes convolucionales. Recuperado de https://ccc. inaoep. mx/~ pgomez/deep/presentations.

¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador. (2020). Retrieved 25 November 2020, from https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

1. ANEXOS

Anexo 1. Arquitecturas.docx

Anexo 2. TERL_CNN Final.ipynb

Anxeo 3. DNNs_InpactorDB FINAL.ipynb