Clasificación de Malware en Android Utilizando Redes Neuronales y Características Dinámicas

Jafeth Calderón Vargas
ECCI
Universidad de Costa Rica
San José, Costa Rica
jafeth.calderon@ucr.ac.cr

Thomas Huber Kelly
ECCI
Universidad de Costa Rica
San José, Costa Rica
thomas.huber@ucr.ac.cr

Jorge Leandro Quirós
ECCI
Universidad de Costa Rica
San José, Costa Rica
jorge.leandro@ucr.ac.cr

Edgar Casasola Murillo
ECCI
Universidad de Costa Rica
San José, Costa Rica
edgar.casasola@ucr.ac.cr

***Resumen*—El estudio y clasificación de malware en Android corresponde a un área de estudio en constante avance. Debido a la adaptación constante del malware para evitar detección, los modelos de detección en Android deben adaptarse para lidiar con protecciones como la ofuscación de código. En este estudio se utilizará el conjunto de datos CCCS-CIC-AndMal-2020 publicado por el Instituto Canadiense de Ciberseguridad para entrenar un modelo de red neuronal que permita clasificar 11 diferentes categorías de malware utilizando características dinámicas del software.**

***Index Terms*—red neuronal, malware, android, clasificación de malware**

I. Introducción

Android corresponde a la principal plataforma que permite utilizar sistemas operativos de alto rendimiento en dispositivos móviles inteligentes. Su cuota de mercado en 2023 se encuentra entre 70.8 % [1] y 71.9 % [2]. El ser la plataforma más utilizada, convierte a estos dispositivos en objetivos prioritarios para el ataque de aplicaciones con software malicioso. En Q1 2023 usuarios del antivirus Kaspersky reportaron 4 948 522 ataques de software malicioso a dispositivos Android [3]. El análisis y clasificación de aplicaciones móviles maliciosas se ha convertido en una prioridad tanto para los usuarios como compañías de seguridad. La clasificación y detección de software malicioso en aplicaciones de Android, generalmente es realizado utilizando aprendizaje de máquinas con análisis de características estáticas y dinámicas [4]. Las características estáticas son extraídas del manifiesto de Android y no requieren que la aplicación se ejecute. Entre estás características se encuentran aspectos como el API, permisos y solicitudes [39]. Las características dinámicas son recolectadas al ejecutar la aplicación en ambientes simulados o teléfonos móviles. Acceso y uso de la batería, cantidad de llamadas del API y transmisión de datos por la red corresponden a ejemplos comunes de características dinámicas [39]. La combinación de características dinámicas y estáticas ha sido efectiva para la clasificación de aplicaciones maliciosas. Sin embargo, debido a que se deben realizar ambos análisis por separado, existe un mayor consumo de recursos y se tarda más en completar el análisis [39]. El utilizar características estáticas tiene la desventaja que los modelos se ven afectados por medidas de seguridad como ofuscación de código y redirecciones de flujo [5]. El uso de modelos de inteligencia artificial como redes neuronales con características dinámicas del software son un siguiente paso que puede generar mejoras en la clasificación y detección de software malicioso en aplicaciones de Android. En este artículo se propone generar un modelo de red neuronal con características dinámicas que permita clasificar 11 familias de software malicioso en aplicaciones de Android. Se utilizará el conjunto de datos CCCS-CIC-AndMal2020 publicado por el Instituto Canadiense de Ciberseguridad [4], [6], [7].

El resto de este artículo se encuentra organizado de la siguiente manera: Sección II se enfoca en el contexto histórico del análisis de aplicaciones maliciosas en Android y también la utilización del conjunto de datos CCCS-CIC-AndMal2020. La sección III se enfoca en el marco teórico y conjunto de datos a utilizar. La sección IV corresponde a la metodología utilizada. Finalmente, las secciones V y VI corresponden a los resultados obtenidos, la discusión y las conclusiones de lo analizado.

II. Antecendentes

El análisis de aplicaciones maliciosas en Android es un campo en constante avance e investigación. El estudio de este problema se ha enfocado en dos áreas: selección de características y generación de modelos con aprendizaje de máquinas. Antes de la publicación de CCCS-CIC-AndMal2020, el enfoque fue la clasificación basada en características individuales como códigos de operación [8], llamadas al API [9], exceso de permisos [10], acceso y conexión a páginas web [11]. Estudios como el de Martín et al. [12] utilizaron cadenas de Markov para clasificar 179 familias de malware. Sihag et al. [13] clasificaron familias de malware utilizando modelos de n-gramas y "hashing" de características. Después de que CCCS-CIC-AndMal2020 fue publicado, múltiples artículos lo han utilizado principalmente como conjunto de pruebas y de validación.

Keyes et al. [4] utilizaron modelos de comportamiento basados en entropía en combinación con Naive Bayes y Random Forest. Utilizando 6 grupos de características dinámicas de malware lograron una precisión y recall del 98 %. Durmus y Varol [14] generaron un modelo estadístico con salidas modeladas utilizando SSPS para clasificar diferentes familias de malware Zhang et al. [15] y Tang et al. [16]. consideraron

el problema que la ofuscación de código tiene sobre la potencial detección de malware y cómo se puede contrarrestar este efecto durante la selección de características para los modelos. En el artículo de Hashem et al. [17] se compararon diferentes métodos como árboles de decisión, Naive Bayes, máquinas de soporte vectorial, regresiones logísticas y vecinos más cercanos para la clasificación de malware utilizando 6 diferentes características. Se encontró que Random Forest fue el mejor método para clasificar malware con una exactitud del 96 %. Islam et al. [18] se realizó un análisis de características dinámicas para malware con un modelo de votos ponderados. Debido a la utilización de modelos de predicción para la selección de características lograron demostrar que un 60 % de las características que se utilizaron por múltiples estudios no eran necesarias para llegar a una clasificación adecuada del malware. Yerima y Khan [19] utilizaron características estáticas para la clasificación y evaluación de diferentes modelos como RF, J48, SVM, NB. Hamood et al. [20] utilizaron algoritmos genéticos para detectar malware en dispositivos Android con conexiones a vehículos de manejo automático.

Las redes neuronales no han sido extensamente utilizadas para la clasificación de malware en combinación con CCCS-CIC-AndMal2020, Rahali et al. [6] utilizaron redes neuronales convolucionales y aprendizaje profundo de imágenes para clasificar malware con una exactitud del 93.36 %. Por otra parte, Atici et al. [21] utilizaron redes neuronales probabilísticas en conjunto con características estáticas y gráficos de control de flujo con el fin de identificar malware.

III. Conjunto de Datos y Marco Teórico

El conjunto de datos fue publicado en 2020 por el Instituto Canadiense de Ciberseguridad en conjunto con el Centro Canadiense de Ciberseguridad [4], [6]. Se recopilaron un total de 200 000 aplicaciones de Android consideradas como programas maliciosos y 200 000 aplicaciones benignas. El software malicioso, se puede describir como una aplicación que tiene como finalidad robar información, dañar y/o causar disrupciones en el funcionamiento correcto de cualquier sistema computacional o dispositivo [22]. En Android, las aplicaciones maliciosas se pueden agrupar en las siguientes categorías:

- Riskware: Programas legítimos que pueden ser utilizados por usuarios maliciosos para causar daño, mediante la eliminación, encriptación, modificación o copia de datos [23].
- Adware: Software que recolecta datos del uso del computador para poder proveer anuncios personalizados al usuario [22]. No siempre es peligroso para el equipo, pero puede causar que el usuario acceda a sitios inseguros que pueden contener otros tipos de malware.
- Troyano: Malware que utiliza el engaño e ingeniería social para hacer que usuarios descarguen y ejecuten software que parece seguro, pero es dañino [24]. Existen diferentes tipos especializados en realizar cierto tipo de tareas como el SMS, los banqueros, los droppers y los spy [25].
- Ransomware: Software malicioso que accede a datos sensibles del usuario y sistema, los encripta y solicita un pago al usuario para desencriptar los archivos afectados [22].
- Potentially Unwanted Apps o PUA: El software potencialmente no deseado corresponde a cualquier software no necesariamente malicioso que cause que el dispositivo se ralentice, despliegue anuncios no deseados o que instale software no deseado sin permiso del usuario [26].
- Backdoor: Métodos mediante el cual un usuario autorizado o no autorizado accede a un sistema con altos permisos burlando el sistema de autenticación y seguridad del sistema [27].
- Scareware: Software maliciosos que engaña y manipula al usuario a visitar paginas o comprar software infectado con otro tipo de malware, mediante el miedo o la intimidación. Esto se logra mediante pop ups donde se le indica al usuario que ha sido infectado y debe seguir una serie de pasos para corregir el problema [28].
- Zero Day: Descripción que se le da a vulnerabilidades de sistemas recién descubiertas que pueden ser explotadas para atacar al sistema [29].

El etiquetado de familias y categorías en el conjunto de datos fue realizado con el software VirusTotal. Se siguió un consenso en donde a cada muestra se le asignó una categoría cuando un 70 % de los antivirus utilizados en la aplicación lo clasificaron en la misma familia y categoría.

En total, el conjunto de datos contiene 14 categorías de programa maliciosos y 191 familias. La distribución de los datos por categoría puede observarse en la Figura 1. El conjunto de datos tiene un muestreo bastante desbalanceado en dónde categorías como Riskware, Adware y Trojans contienen entre 2 y 3 veces más datos que el resto de las categorías. El Sobre Muestro Sintético de la Minoría, o SMOTE por sus siglas en inglés, es una técnica para lidiar con problemas de desbalanceo de clases en un conjunto de datos. Se busca crear muestras sintéticas de las clases con menor cantidad, tomando los k vecinos más cercanos de una muestra aleatoria [34].

Por otra parte, el conjunto de datos recompila información de características estáticas y dinámicas para el software. Las características dinámicas fueron evaluadas y registradas utilizando un ambiente emulado en lugar de dispositivos Android físicos [4]. En total se registraron un total 141 características dinámicas separadas en 6 categorías [7]:

- Memoria: Uso de memoria con base en las operaciones que se llevan a cabo. Por ejemplo, proporciones de memoria privada y compartida, cantidad de veces que se colocan datos en la pila y tamaño general de la pila.
- API: Número de llamados a API en el sistema. Se consideraron llamadas de API a procesos de Android, información del dispositivo, manejo de contraseñas y cuentas, acceso y uso de mensajería de texto.
- Red: Cantidad de datos transmitidos y recibidos en bytes o paquetes a través de la red.
- Batería: Consumo de batería y accesos emitidos durante

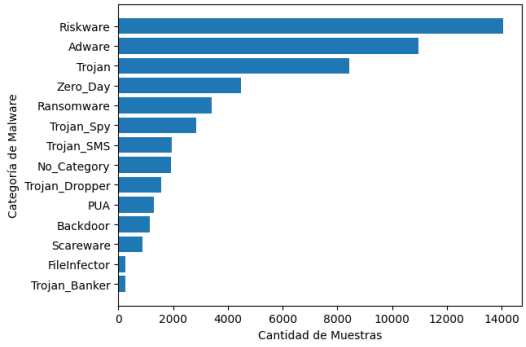


Figura 1. Conteo de muestras por cada categoría registrada en CCCS-CIC-AndMal-2020.

la ejecución.

- Registros: Registros en bitácora del total de acciones realizadas por la aplicación.
- Procesos: Número de procesos en el sistema y las interacciones del software malicioso con los mismos.

Debido a la alta densidad de datos provenientes de un análisis dinámico de una aplicación, es necesario utilizar métodos de reducción de dimensionalidad o selección de características. Entre las técnicas para realizar esto está la de bosque aleatorio. En este caso cada árbol de decisión del bosque prueba una serie de características combinadas y determina cuáles tienen el mayor impacto y grado de pureza al momento clasificar [33].

El conjunto de datos se ha utilizado para entrenar modelos de aprendizaje de máquinas y, recientemente, como se pudo observar en los antecedentes, se comenzó a utilizar aprendizaje profundo como las redes neuronales. Ésta es una popular técnica de aprendizaje profundo donde cada neurona recibe una serie de entradas y se le asocia un peso a cada entrada, de modo que realiza una suma ponderada y pasa el resultado por una función de activación que determina la salida de dicha neurona [30]. Es decir, para determinar si transmite datos a la siguiente capa o no. Dependiendo de sus capas y el funcionamiento de la red, estás se pueden clasificar en diferentes arquitecturas. Una de las arquitecturas más comunes son las redes neuronales completamente conectadas que consisten en un conjunto de capas de neuronas en las que todas las neuronas de una capa conectan con todas las de la capa siguiente [31].

Como se mencionó anteriormente, las funciones de activación determinan la salida de una neurona con base en sus entradas. Se propone el uso de la función ReLU la cual se utiliza para mejorar el rendimiento del aprendizaje de la red. Su fórmula está dada por:

$$R(x) = Max(0, x) \tag{1}$$

Por otra parte, la función de activación softmax se utiliza en la capa final o de salida de las redes de clasificación multiclase. Esta genera las probabilidades de que una entrada pertenezca a cada una de las clases disponibles para clasificación. La función correspondiente es:

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2}$$

Otro aspecto importante de las redes neuronales corresponde a la validación de los resultados con el fin de optimizar los hiper parámetros de la red. La validación cruzada con k iteraciones es una técnica para evaluar el rendimiento del modelo, en donde los datos de prueba se dividen en k subconjuntos y se entrena el modelo con $k-1$ subconjuntos, probando en el subconjunto restante, y de esta misma forma para cada combinación de subconjuntos con el fin de ajustar los híper parámetros de la red [31].

IV. Metodología

Para poder aprender las relaciones que existen entre las características de los datos y clasificar las muestras de programas malignos, se propone la implementación de selección de características y un modelo de aprendizaje profundo. Inicialmente se realiza el procesamiento de los datos para delimitar el alcance del problema, selección de características y sobremuestreo sobre el conjunto de entrenamiento para tratar el desbalance. Una vez se encuentra la mejor arquitectura e híper parámetros, se entrena el modelo final y se evalúa sobre el conjunto de pruebas. Finalmente, se obtienen promedios de métricas para las predicciones realizadas.

IV-A. Preprocesamiento de Datos

Con respecto a las muestras de cada categoría, en la Figura 1 se observa la distribución de los datos para cada una. Tanto para FileInfector como para Banker se tienen menos de 300 muestras, por lo que en comparación con las 14,000 aproximadamente que se tienen para Riskware, no constituyen una cantidad razonable de datos. Incluso al aplicar técnicas de sobre muestreo para estos dos casos, no se alcanzan buenos resultados por el desbalance que existe entre los datos, por ello se decide descartar ambas categorías. Adicionalmente se descartan aquellos datos sin una categoría específica, ya que se sale del enfoque de clasificación que se desea desarrollar.

IV-B. Selección de Características

Se emplea el método de bosque aleatorio. Se encuentra que aquellos atributos que corresponden a eventos de interacción con la bitácora, número de interacciones con procesos, grabación multimedia, localización, notificación, información del Wifi, uso de bases de datos, detención de procesos, entre otros, no aportan información a la clasificación, por lo que se descartan del conjunto de datos, efectivamente reduciendo su dimensionalidad.

Por otra parte, entre las características que más aportan información para realizar la clasificación están la cantidad de veces que se realizan llamados a crear procesos, abrir conexiones, detener servicios, eventos para obtener información del servicio de telefonía, ejecutar procesos, eliminar archivos, registrar observadores de contenido para identificar cambios en ciertos datos, cantidad de bytes transmitidos totales y

paquetes recibidos totales, manipulación de la batería, uso de memoria, entre otros. Estos eventos capturados se relacionan con los diferentes tipos de malware que se quieren clasificar por lo que tiene sentido que sean aquellos que determinan si cierto programa cabe dentro de una categoría u otra según la cantidad de veces que se realice o las cantidades registradas para los diferentes contadores. Es importante considerar que al tratarse de la técnica de bosque aleatorio, es posible que entre una y otra ejecución de la clasificación, se tomen otras características y se les asigne diferentes índices de ganancia de información por la naturaleza aleatoria de la técnica de aprendizaje automático. Sin embargo, las características que se mencionan se mantienen por lo general siempre muy por encima del resto que presentan ganancia de información en una menor proporción, por lo que se puede inferir que estos son los indicadores que por lo general se emplean para clasificar malware según el comportamiento de las diferentes categorías.

IV-C. Sobremuestreo

Por el desbalance observado anteriormente, se decide aplicar la técnica de SMOTE para realizar un sobre muestreo para las 11 categorías restantes sobre el conjunto de datos de entrenamiento. Esto mejora la capacidad de generalización para las categorías con menos muestras inicialmente.

IV-D. Arquitectura de la Red Neuronal

La arquitectura seleccionada para la red neuronal se compone de las siguientes capas:

- Capa de Entrada: Las características obtenidas según la selección del bosque aleatorio.
- Capas Ocultas: Función de activación ReLU, sesgos inicializados en cero, inicializador de pesos con algoritmo He normal, dropout con probabilidad de 0.15, excepto en la primera capa oculta.
  - Capa Densa 1: 120 neuronas.
  - Capa Densa 2: 80 neuronas.
  - Capa Densa 3: 50 neuronas.
  - Capa Densa 4: 30 neuronas.
  - Capa Densa 5: 20 neuronas.
- Capa de Salida: Función de activación softmax, sesgos inicializados en cero, inicializador de pesos con algoritmo Glorot normal.
  - Capa Densa 6: 11 neuronas.

En cuanto al optimizador, se utiliza Adam con valor por defecto de tasa de aprendizaje de 0.001. Como función de pérdida se elige la entropía cruzada categórica en su versión dispersa. Como métrica de entrenamiento se visualizan la función de pérdida y la exactitud categórica dispersa que se adecúa más a modelos de clasificación multiclase.

Se emplea validación cruzada con 10 iteraciones sobre el conjunto de datos de pruebas para el ajuste de estos híper parámetros. Se ejecuta el modelo por 100 épocas debido al lento aprendizaje por el que se opta, esto para propiciar la generalización, sin embargo, para compensar, se elige un tamaño de lote de 256 para aprovechar los recursos computacionales.

V. Resultados y Discusión

En la Figura 2 se puede observar la exactitud categórica global del modelo para cada época. Se debe considerar que se aplica el sobremuestreo sobre los datos de entrenamiento por lo que tiene mejores probabilidades de clasificar los datos sintéticos que ya ha visto, lo que se traduce en mejores métricas.

En el Cuadro I se pueden observar los datos de precisión, sensibilidad, el puntaje F1 y la cantidad de muestras en el conjunto de pruebas para cada categoría. Para promediar estos resultados se utiliza el macro promedio que da el mismo peso a todas las categorías y el ponderado que toma en cuenta la cantida de muestras en cada una.

Es interesante observar que existen diferencias significativas entre varias categorías. De manera generar se puede observar que en el modelo:

- Riskware presenta la mayor precisión.
- Zero Day presenta la menor precisión.
- Trojan Spy presenta la mayor sensibilidad.
- Zero Day presenta la menor sensibilidad.

Los resultados obtenidos no son completamente inesperados. Pese a que se esperaba un modelo con una mayor sensibilidad, el propuesto presenta limitaciones al clasificar ciertas categorías como Zero Day y Trojan Dropper. Esto puede explicarse considerando dos factores: naturaleza del área estudiada y las limitaciones de los datos utilizados.

La clasificación de malware corresponde a un área de constantes avances tanto para el malware como para los modelos de detección [4]. Estudios han demostrado que generalmente las métricas estáticas funcionan mejor para clasificar malware [17], mientras que métricas dinámicas son más resilientes a técnicas de evasión y ofuscación de código [15], [16]. Lo anterior nos puede dar un indicio de porqué el modelo no alcanza sensibilidades realmente altas, nuestro modelo se basa únicamente en métricas dinámicas.

Es importante considerar también el efecto que tiene la similitud en el comportamiento del malware, un ejemplo en los resultados corresponde a Trojan Dropper y Ransomware. La Figura 3 muestra que 75 de los casos en Trojan Dropper

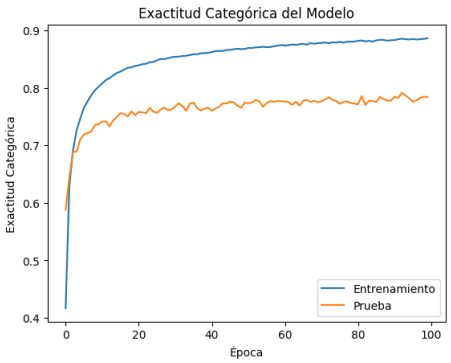


Figura 2. Exactitud global del modelo para conjuntos de entrenamiento y pruebas.

Cuadro I
RESULTADOS DE VALIDACIÓN

Categoría	Precisión	Sensibilidad	Puntaje F1	Soporte
Riskware	0.93	0.77	0.84	4216
Adware	0.83	0.81	0.82	3294
Trojan	0.86	0.80	0.83	2531
Zero Day	0.48	0.66	0.55	1343
Ransomware	0.83	0.75	0.79	1023
Trojan Spy	0.86	0.89	0.87	852
Trojan SMS	0.59	0.81	0.68	582
Trojan Dropper	0.63	0.74	0.68	471
PUA	0.72	0.80	0.76	387
Backdoor	0.60	0.77	0.68	341
Scareware	0.71	0.85	0.77	266
Macro	0.73	0.79	0.75	15306
Ponderado	0.81	0.78	0.79	15306

fueron catalogados incorrectamente como Ransomware, y 116 en el sentido contrario. Esto no es completamente sorprendente ya que en muchas ocasiones el Ransomware utiliza Trojans como vehículo para la infección [35], por lo que es de esperar comportamientos similares. Otros estudios han utilizado combinaciones de características estáticas y dinámicas con el fin de mejorar el poder predictivo de los modelos. Esta modificación corresponde a una potencial mejora para la investigación. Al combinar características dinámicas y estáticas se generan modelos resilientes a técnicas de evasión como ofuscación de código pero que tienen un poder predictivo mayor [5]. También es importante considerar la arquitectura de la red neuronal. La adición de capas convoluciones [6] o modificar la red para utilizar un modelo de redes neuronales probabilísticas [21] pueden mejorar el poder predictivo de la red. De igual forma se debe considerar que por la naturaleza del análisis dinámico de malware, es común encontrar diversas mediciones con valores de 0. Incluso después de ser normalizados, si múltiples características presentan valores nulos, estos podrían causar problemas debido a que las capas ocultan utilizan una función ReLU, lo que lleva a considerar otras funciones de activación.

Por otra parte, limitaciones en los datos utilizados son potencialmente otro factor que afecta la capacidad predictiva del modelo. Categorías como Riskware y Adware tienen casi 3 veces más datos que categorías como Zero Day. Esta disparidad en la cantidad de datos de muestreo tiene efectos negativos en los modelos de predicción [36].

El efecto de este desbalance en los datos es justamente uno de los principales factores por los que Zero Day es la clase con la sensibilidad y precisión más bajas. Una clasificación incorrecta en el conjunto de pruebas para Zero Day tiene un efecto mucho mayor sobre las métricas evaluadas que para Riskware. A manera de mejora, se podría combinar la información con otro conjunto de datos que permita obtener muestras más variadas para los malwares menos representados.

Otro potencial efecto sobre los resultados corresponde a la manera en que fue etiquetado el conjunto de datos. Para el etiquetado de los datos se utilizó VirusTotal [6]. Esta aplicación utiliza un consenso de antivirus, en dónde la etiqueta

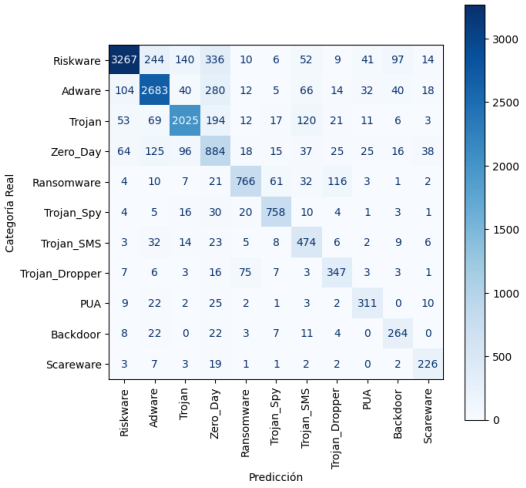


Figura 3. Matriz de confusión con conteos crudos de datos en el conjunto de validación.

es asignada si al menos un 70 % acuerda que el malware corresponde a dicha categoría. Esto naturalmente introduce incertidumbre en el sistema, ya que incrementa la probabilidad de una categorización errónea para los datos.

Este fenómeno ha sido estudiado en otras áreas, en dónde clasificaciones ambiguas tienen efectos negativos sobre el poder predictivo de modelos de inteligencia artificial [37], [38]. En la Figura 3 se notan 120 de los Trojan que fueron clasificados como Trojan SMS, disminuyendo la precisión de esta última categoría. Una potencial mejora al estudio corresponde a cambiar el método de etiquetado de los datos, ya sea modificando el umbral a utilizar para colocar la etiqueta o crear etiquetas más generales que introduzcan menos incertidumbre.

VI. CONCLUSIONES

En el área de clasificación de malware en Android mediante aprendizaje profundo se utilizan diversas arquitecturas que presentan rendimientos variados. En este artículo se propone el uso de una red neuronal completamente conectada sencilla para la clasificación de malware en 11 categorías según características dinámicas. Se encuentra que para el conjunto de datos elegido se tienen muestras desbalanceadas por lo que esto afecta al aprendizaje de la red y a la evaluación del modelo entrenado. El sobremuestreo mejora la generalización de las categorías, sin embargo, a la hora de evaluar los datos la precisión se ve afectada. Otro aspecto a tomar en cuenta es el etiquetado de los datos, ya que solamente un 70 % de los antivirus concordaron sobre la categoría correspondiente. Finalmente, se determina que podrían utilizarse otras arquitecturas más complejas para una mejor clasificación, no obstante, las métricas obtenidas son aceptables para el contexto del conjunto de datos utilizado y para el poco requerimiento computacional del modelo propuesto.

Referencias

[1] P. Taylor, "Global mobile OS market share 2023," Statista, https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009 (accessed Jul. 8, 2023).

[2] C. Renée, "Android market share vs. Apple: everything you want to know," History-Computer, https://history-computer.com/android-market-share-vs-apple-everything-you-want-to-know/ (accessed Jul. 8, 2023).

[3] A. Kivva, "Smartphone malware statistics, Q1 2023," Securelist English Global, https://securelist.com/it-threat-evolution-q1-2023-mobile-statistics/109893/ (accessed Jul. 8, 2023).

[4] D. S. Keyes et al., "EntropLyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics," 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), 2021. doi:10.1109/rdaaps48126.2021.9452002

[5] V. Sihag, M. Vardhan, P. Singh, and G. Choudhary, "De-LADY: Deep learning based Android malware detection using dynamic features," Journal of Internet Services and Information Security (JISIS), vol. 11, no. 2, pp. 34–45, May 2021. doi:10.22667/JISIS.2021.05.31.034

[6] A. Rahali et al., "DIDroid: Android malware classification and characterization using deep image learning," 2020 the 10th International Conference on Communication and Network Security, 2020. doi:10.1145/3442520.3442522

[7] Canadian Institute for Cybersecurity, "CCCS-CIC-AndMal-2020," Canadian Institute for Cybersecurity — UNB, https://www.unb.ca/cic/datasets/andmal2020.html (accessed Jul. 9, 2023).

[8] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—a state of the Art Survey," ACM Computing Surveys, vol. 52, no. 5, pp. 1–48, 2019. doi:10.1145/3329786

[9] S. Alam, S. A. Alharbi, and S. Yildirim, "Mining nested flow of dominant apis for detecting Android malware," Computer Networks, vol. 167, p. 107026, 2020. doi:10.1016/j.comnet.2019.107026

[10] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Scalable and robust unsupervised Android malware fingerprinting using community-based network partitioning," Computers & Security, vol. 96, p. 101932, 2020. doi:10.1016/j.cose.2020.101932

[11] S. Wang et al., "Deep and broad URL feature mining for Android malware detection," Information Sciences, vol. 513, pp. 600–613, 2020. doi:10.1016/j.ins.2019.11.008

[12] A. Martín, V. Rodríguez-Fernández, and D. Camacho, "Candyman: Classifying android malware families by modelling dynamic traces with Markov chains," Engineering Applications of Artificial Intelligence, vol. 74, pp. 121–133, 2018. doi:10.1016/j.engappai.2018.06.006

[13] V. Sihag, A. Mitharwal, M. Vardhan, and P. Singh, "Opcode N-gram based malware classification in Android," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020. doi:10.1109/worlds450073.2020.9210386

[14] O. Durmus and A. Varol, "Analysis and modeling of cyber security precautions," 2021 9th International Symposium on Digital Forensics and Security (ISDFS), 2021. doi:10.1109/isdfs52919.2021.9486345

[15] X. Zhang, F. Breitinger, E. Luechinger, and S. O'Shaughnessy, "Android Application Forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations," Forensic Science International: Digital Investigation, vol. 39, p. 301285, 2021. doi:10.1016/j.fsidi.2021.301285

[16] J. Tang, R. Li, Y. Jiang, X. Gu, and Y. Li, "Android malware obfuscation variants detection method based on multi-granularity opcode features," Future Generation Computer Systems, vol. 129, pp. 141–151, 2022. doi:10.1016/j.future.2021.11.005

[17] A. Hashem, A. El Shenawy, and M. Ashraf, "Android malware category and family detection and identification using machine learning," CoRR, vol. 2107, no. 2107.01927, 2021. doi:10.1145/3442520.3442522

[18] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, "Android malware classification using optimum feature selection and ensemble machine learning," Internet of Things and Cyber-Physical Systems, vol. 3, pp. 100–111, 2023. doi:10.1016/j.iotcps.2023.03.001

[19] S. Y. Yerima and S. Khan, "Longitudinal performance analysis of machine learning based Android malware detectors," 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2019. doi:10.1109/cybersecpods.2019.8885384

[20] L. Hammood, İ. A. Doğru, and K. Kılıç, "Machine learning-based adaptive genetic algorithm for Android malware detection in auto-driving vehicles," Applied Sciences, vol. 13, no. 9, p. 5403, 2023. doi:10.3390/app13095403

[21] M. A. Atici, S. Sagiroglu, and I. A. Dogru, "Android malware analysis approach based on control flow graphs and machine learning algorithms," 2016 4th International Symposium on Digital Forensic and Security (ISDFS), 2016. doi:10.1109/isdfs.2016.7473512

[22] "What is malware?," Cisco, https://www.cisco.com/site/us/en/products/security/what-is-malware.html (accesado Jul. 8, 2023).

[23] "¿Qué es el riskware?," latam.kaspersky.com, https://latam.kaspersky.com/resource-center/threats/riskware (accesado Jul. 8, 2023).

[24] "Trojan horse – Virus or malware?," Malwarebytes, https://www.malwarebytes.com/trojan (accessed Jul. 8, 2023).

[25] What is a Trojan horse and what damage can it do?," www.kaspersky.com, https://www.kaspersky.com/resource-center/threats/trojans (accessed Jul. 8, 2023).

[26] "Block potentially unwanted applications with Microsoft defender antivirus," Block potentially unwanted applications with Microsoft Defender Antivirus — Microsoft Learn, https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/detect-block-potentially-unwanted-apps-microsoft-defender-antivirus (accesado Jul. 8, 2023).

[27] "Backdoor computing attacks," Malwarebytes, https://www.malwarebytes.com/backdoor (accesado Jul. 8, 2023).

[28] S. Bagui and H. Brock, "Machine learning for android scareware detection," Journal of Information Technology Research, vol. 15, no. 1, pp. 1–15, Mar. 2022. doi:10.4018/jitr.298326

[29] "What is a Zero-day Attack? - Definition and Explanation," www.kaspersky.com, https://www.kaspersky.com/resource-center/definitions/zero-day-exploit (accessed Jul. 8, 2023). https://www.unb.ca/cic/dataconjuntos/andmal2020.html

[30] "What are neural networks?," IBM, https://www.ibm.com/topics/neural-networks (accessed Jul. 8, 2023).

[31] M. Ekman, Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow. Boston: Addison-Wesley, 2021.

[32] R. Wambui, "Cross-Entropy Loss and Its Applications in Deep Learning," neptune.ai, https://neptune.ai/blog/cross-entropy-loss-and-its-applications-in-deep-learning (accessed Jul. 8, 2023).

[33] R.-C. Chen, C. Dewi, S.-W. Huang, and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," Journal of Big Data, vol. 7, no. 1, Jul. 2020. doi:10.1186/s40537-020-00327-4

[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, Jun. 2002. doi:10.1613/jair.953

[35] Kaspersky, "Ransomware attacks and types – how encryption trojans differ," www.kaspersky.com, https://www.kaspersky.com/resource-center/threats/ransomware-attacks-and-types (accessed Jul. 9, 2023).

[36] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in Convolutional Neural Networks," Neural Networks, vol. 106, pp. 249–259, 2018. doi:10.1016/j.neunet.2018.07.011

[37] J. Walsh-Messinger et al., "Relative importance of symptoms, cognition, and other multilevel variables for psychiatric disease classifications by machine learning," Psychiatry Research, vol. 278, pp. 27–34, 2019. doi:10.1016/j.psychres.2019.03.048

[38] M. Walter et al., "Translational machine learning for psychiatric neuroimaging," Progress in Neuro-Psychopharmacology and Biological Psychiatry, vol. 91, pp. 113–121, 2019. doi:10.1016/j.pnpbp.2018.09.014

[39] R. R. C. A. K. S., "A Preface on Android Malware: Taxonomy, Techniques and Tools", IJRITCC, vol. 5, no. 6, pp. 1111 –, Jun. 2017.