An Assessment of the Effectiveness of Pretrained Neural Networks for Malware Detection

Eduardo Malvacio and Julio Cesar Duarte

***Abstract*—The speed at which malware develops is much faster than analysts are able to analyze it, and it is now a dangerous threat to businesses, critical infrastructure and individuals, forcing anti-virus companies to develop fast and reliable methods for detecting malware. Transfer learning, a deep learning technique, has special abilities such as training speed, lower training dataset size requirements and reduced demand for domain expert knowledge. In this paper, we compared the prediction and model generation performances by using modern convolutional neural networks (Resnet, DenseNet, VGG and AlexNet), which have proven to be successful in the visual classification problem for malware detection. In addition, a dataset containing 14,972 malware samples and 14,500 samples of benign files with some function similar to that possessed by malwares was proposed and used in the study. The best accuracy obtained was 96.77 % corresponding to the *DenseNet-201* architecture.**

***Index Terms*—Malware, malware detection, convolutional neural networks, computer vision, transfer learning.**

I. Introducción

Un software malicioso que se instala en un dispositivo sin el consentimiento del usuario es denominado malware. Este puede robar información, dañar el sistema computacional o simplemente mostrar publicidad no deseada. A medida que los sistemas se han vuelto más complejos y conectados, el malware se ha adaptado para beneficiarse con las nuevas tecnologías.

La detección de malware se ha convertido en uno de los campos de investigación más importantes. De acuerdo con el tipo de malware, puede ser realizado análisis de tipo estático o dinámico [1].

Otra alternativa para identificar el malware es el enfoque de visualización. En trabajos recientes, ese enfoque de análisis [2], [3] se ha utilizado como una solución eficaz para identificar software malicioso, ya que analiza el archivo ejecutable. La visualización de software malicioso es un método en el que se representa software malicioso como una imagen mediante la extracción de sus binarios [4]. En lugar de analizar el código, se analiza la textura de la imagen generada por la representación del malware, lo que permite que el clasificador identifique el malware, incluso si el atacante ha utilizado técnicas de ofuscación o modificación [2].

Ante la imposibilidad de analizar todas las muestras de malware que son descubiertas diariamente es necesario utilizar técnicas de clasificación de malware, que sean capaces de seleccionar un subconjunto de muestras que merecen el análisis manual por parte de un analista humano, ya que no todas las muestras de malware son iguales ni presentan los mismos comportamientos.

Este trabajo pretende identificar archivos maliciosos utilizando un conjunto de datos de archivos benignos con funciones similares a la de los malwares, con el fin de optimizar la selección de muestras que un analista humano debería analizar. Para realizar dicha tarea se optó por convertir las muestras en imágenes y realizar una clasificación visual de las mismas. Además se realizará un estudio comparativo entre varias redes preentrenadas para comprobar si el rendimiento de dichos clasificadores es comparable al de los clasificadores creados específicamente para el problema.

Como principales contribuciones se pueden enumerar:

- Crear un *benchmark* mediante una base de datos con archivos maliciosos y benignos representativos de la tarea que se está modelando.
- Construir una base de datos de archivos benignos con funciones similares a los archivos maliciosos, posibilitando disminuir la brecha para la identificación.
- Obtener conclusiones sobre la viabilidad de utilizar redes neuronales preentrenadas para su uso en la clasificación de imágenes de malware y otros problemas similares.

El presente trabajo consta de 6 secciones; en la sección II, se presentan los trabajos relacionados, exponiendo las diferencias y similitudes entre ellos. La metodología empleada se puede apreciar en la sección III. En la sección IV se muestra el conjunto de datos y su diferenciación con otros trabajos que utilizan conjuntos de datos estándares, además de una redefinición de los términos malware y benigno. En la sección V se describen los experimentos realizados finalizando con una tabla comparativa de los 8 modelos experimentados. Por último, en la sección VI se exponen las conclusiones del presente trabajo.

II. Trabajos relacionados

Los investigadores han estado estudiando el malware para comprender mejor su comportamiento y estructura. Sin embargo, para clasificarlos de manera más precisa y eficiente, es necesario contar con la ayuda de la inteligencia artificial. Esta sección presenta los trabajos relacionados con la clasificación de malware basada en el análisis estático, dinámico, el aprendizaje automático y el uso del aprendizaje profundo.

En la actualidad existen dos técnicas principales de extracción de características para la detección de malware: el análisis estático y el análisis dinámico. El análisis estático es un proceso en el que se examina el código fuente de un

Eduardo Malvacio and Julio Cesar Duarte are with Seção de Ensino de Engenharia de Computação, Instituto Militar de Engenharia - Urca, Rio de Janeiro - RJ, Brazil, e-mail:edumalvacio@ime.eb.br

programa para identificar las características y funciones del mismo. Esto se hace sin ejecutar el programa, lo que significa que el analista de malware debe tener un buen conocimiento de cómo funcionan los lenguajes de programación y los sistemas operativos. Por otro lado, el análisis dinámico es una técnica de análisis de seguridad utilizada para examinar el comportamiento de un programa sospechoso.

Yang *et al.* [5] propusieron un método de análisis estático mediante un gráfico de llamadas a funciones, un trabajo similar realizaron Nar *et al.* [6] al extraer características del archivo portable ejecutable (PE) del malware, y las secuencias de opcode. Alsoghyer *et al.* [7] y Faris *et al.* [8] implementaron el análisis estático para extraer varias características estáticas de los archivos binarios de malware de Android, como los permisos y las llamadas a la API, posteriormente, realizaron técnicas de aprendizaje automático para detectar aplicaciones maliciosas. Jeon *et al.* [9] realizaron un análisis estático para la detección de malware utilizando *Tensorflow*.

En cuanto al análisis basado en análisis dinámico, se obtienen las características de comportamiento del malware, como las llamadas a APIs, las actividades en la red y los archivos de registro, entre otros.

Mohaisen *et al.* [10] desarrolló un esquema automatizado de malware y etiquetado. En el sistema propuesto, se extrajeron varias características basadas en el comportamiento durante el análisis dinámico, como las actividades de la red, los sistemas de archivos y los registros. Sihwail *et al.* [11] utilizaron técnicas de análisis forense de memoria con un enfoque de análisis dinámico. Primero se extrajeron los artefactos maliciosos de la memoria y después se utilizó *Cuckoo Sandbox* para monitorizar el comportamiento del malware durante su ejecución. Por último, los artefactos maliciosos y el informe de comportamiento se combinaron para crear el conjunto de datos de características para su posterior clasificación. Sin embargo, el software malicioso puede cambiar su comportamiento durante la ejecución en un entorno virtual; por lo tanto, el análisis basado en características dinámicas puede no capturar el comportamiento real del software malicioso.

Recientemente, se ha llevado a cabo una amplia investigación sobre la clasificación de malware mediante la aplicación del enfoque basado en la visualización [12]–[17].

Algunos autores han desarrollado soluciones CNN, en las que no han utilizado ningún modelo previamente entrenado. Moussas *et al.* [13] desarrollaron un sistema de clasificación de malware visualizado basado en una red neuronal artificial (RNA). El sistema de clasificación propuesto utiliza las características extraídas de la base de datos Malimg para entrenar la RNA. Posteriormente, el modelo entrenado se utiliza en la clasificación de diferentes muestras de la base de datos Malimg. La exactitud de los resultados obtenidos usando una capa oculta fue del 96 %, por otro lado, la implementación utilizando dos capas ocultas alcanzó una exactitud del 99,135 %. Roseline *et al.* [12] utilizó el aprendizaje por transferencia y ajuste fino para introducir la detección eficiente de malware en el contexto de familias desequilibradas sin necesidad de procesos complejos de extracción de características o aumento de datos, logrando una exactitud del 99,97 %.

Gibert *et al.* [15] convirtieron el malware en imágenes en escala de grises para desarrollar un esquema de aprendizaje profundo basado en una CNN. El esquema propuesto extrae patrones para clasificar el software malicioso. Además de los patrones, se pueden implementar diferentes funciones en el proceso de visualización de malware.

Xiao *et al.* [18] han demostrado que la combinación de una CNN profunda con un gráfico de entropía ayuda a mejorar el proceso de clasificación de patrones de malware.

Vasan *et al.* [17] también utilizaron imágenes a color en las que implementaron el IMCFN, un sistema de clasificación de malware basado en imágenes utilizado una CNN *Fine-tuned*. Inicialmente, convirtieron el malware en imágenes a color utilizando un algoritmo de mapa de colores. Para superar el problema del desequilibrio del conjunto de datos aplicaron técnicas de aumento de datos durante el proceso de ajuste. Además, también compararon la aplicación de la clasificación entre imágenes en escala de grises y en color, y obtuvieron el mismo resultado al tener un mejor rendimiento de clasificación en caso de utilizar imágenes en color. Aunque el IMCFN logró una exactitud del 98,82 % en el conjunto de datos Malimg, el sistema propuesto tiene una complejidad adicional debido a las técnicas de aumento utilizadas y al algoritmo de mapa de colores.

Algunos investigadores optaron por combinar varios modelos de entrenamiento para mejorar el proceso de clasificación visual [14], [19], [20]. En Nisa *et al.* [14], el *Scale Feature Texture Analyzer* (SFTA) se combina con dos modelos de técnicas de CNN profundas (DCNN), AlexNet e Inception-v3, para mejorar la exactitud de la detección de malware.

Vasan *et al.* [20] utilizan VGG16 y ResNet50 para la extracción de características, sin embargo, en el conjunto de arquitecturas de CNN, se han implementado dos clasificadores SVMs, SoftMax y Multiclass, luego, se aplicó un proceso de Análisis de Componentes Principales (PCA) para disminuir la dimensionalidad de las características, mientras que se utilizó un proceso de fusión antes del proceso de clasificación. Además, las CNN pueden combinarse con otras técnicas, por ejemplo, la solución propuesta por Narayanan *et al.* [21] implementó un esquema de clasificación basado en redes neuronales recurrentes y convolucionales utilizando imágenes de malware.

En otros escenarios, la CNN puede aplicarse para extraer características mientras se implementan técnicas de aprendizaje automático en el proceso de clasificación del malware [22]–[24]. En Roseline *et al.* [22], el malware se clasificó implementando la técnica de comité *Random Forest* secuencialmente en múltiples capas. La solución propuesta se realiza en dos pasos. Inicialmente, se analizan las características en bruto utilizando diferentes tamaños de ventana. Posteriormente, se aplican técnicas diferentes de clasificación, entre ellas, *Bosque aleatorio* (RF), *Clasificador de árbol extra* (ETC) y *Regresión logística* (LR).

Ouahab *et al.* [23] desarrollaron un descriptor de características de imagen para extraer las similitudes entre las imágenes de malware. Luego, implementaron el algoritmo *K-Nearest Neighbor* (KNN) para realizar el proceso de clasificación.

Otro enfoque interesante es el de Deore *et al.* [25] que proponen el uso de una técnica de visualización en la que el

código de malware desensamblado se convierte en imágenes grises, así como el uso de parámetros estadísticos basados en similitud de imagen (ISSP).

En el trabajo de Mohino *et al.* [26] se propuso una metodología para el análisis de malware en el sistema operativo Linux, que es un campo tradicionalmente pasado por alto en comparación con los otros sistemas operativos. La metodología propuesta es probada por un malware específico de Linux, y los resultados de las pruebas obtenidas tienen una alta efectividad en la detección de malware.

En Dhalaria *et al.* [27], los autores han presentado un enfoque híbrido para la detección y clasificación de malware para Android. Emplearon varios algoritmos de clasificación en las características estáticas y dinámicas obtenidas para detectar malware. Los hallazgos demuestran que un *Random Forest* proporcionó mejores resultados para la detección y clasificación de malware

Por último, Naeem *et al.* [28] diseñaron un método para convertir archivos binarios de malware en imágenes en escala de grises y utilizó dos patrones, local y global (LGMP), para clasificar el malware. De la parte experimental de su estudio, la exactitud de la clasificación alcanzó el 98,4 %.

Aunque se han propuesto muchos algoritmos que utilizan la visualización de imágenes de malware, no se encontró ninguno que haya utilizado muestras de archivos benignos con funciones similares a la de los malwares. Los métodos utilizados hasta la actualidad son útiles para la detección de archivos maliciosos cuyo comportamiento dista mucho del comportamiento de los archivos benignos, sin embargo dado que la cantidad de analistas es significativamente menor a la de la aparición de nuevos malwares se hace relevante la necesidad de mejorar la preclasificación del malware que los analistas deben analizar manualmente.

El presente trabajo pretende abordar, utilizando redes neuronales preentrenadas y transferencia de aprendizaje, el problema de la identificación de malware cuando las muestras de malware y las muestras de archivos benignos comparten funciones similares. Visto de otra forma, podría decirse que las imágenes de las muestras de malware son muy similares a las imágenes de los archivos benignos, lo que aumenta la dificultad en la clasificación. Siendo esperable una exactitud menor a la de los trabajos recientemente expuestos.

III. Metodología

El aprendizaje por transferencia es una solución adecuada para la clasificación de imágenes, ya que proporciona una metodología mediante la cual el conocimiento puede transferirse desde el dominio de origen a un dominio de destino de un problema similar con pocas personalizaciones en los modelos profundos de CNN.

Los experimentos fueron divididos en seis fases: recolección de muestras, preprocesamiento de muestras, generación de imágenes, generación de modelos, entrenamiento de modelos y clasificación de muestras, puede verse un modelo con las seis fases diagramadas en la Fig. 1. La división del problema general en fases o etapas resultó indispensable para poder abordar problemas más pequeños y complejos con una mayor

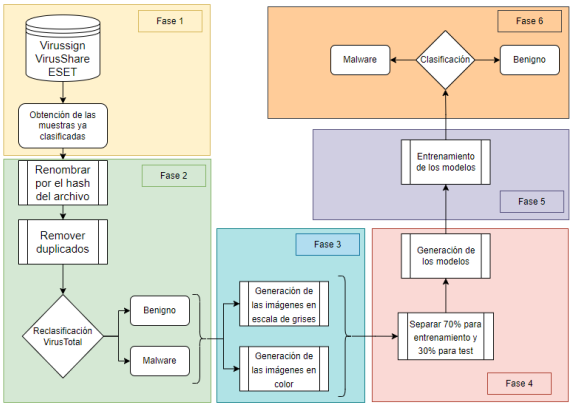

Fig. 1. Esquema del método propuesto. Fuente: [Autoría propia].

eficiencia. La separación de las fases del problema general, permitió analizar cada uno de los subproblemas de forma independiente, y así poder identificar y resolver con mayor facilidad los errores en cada uno de ellos.

Durante la fase 1 (recolección de muestras) se obtuvieron las muestras de malware de dos fuentes diferentes: VirusShare [29] y VirusSign [30]. Por otra parte las muestras benignas fueron donadas por la empresa ESET [31].

Como las muestras procedían de fuentes diferentes, a fin de eliminar los duplicados, y con la finalidad de reclasificar las muestras de malware y archivos benignos se realizó la fase 2 (preprocesamiento de muestras), en esta fase se renombraron los archivos, se eliminaron los duplicados y se reclasificaron las muestras.

Posteriormente durante la fase 3 (generación de imágenes) se generaron imágenes de todas las muestras resultantes de la fase anterior, se crearon imágenes en escalas de grises e imágenes a color de cada muestra.

Durante la fase 4 (generación de modelos) se realizó la separación de las muestras en dos conjuntos: *entrenamiento* y *test* y posteriormente se crearon los 8 modelos de redes neuronales preentrenadas (ver Tabla II).

La fase que más tiempo y esfuerzo demandó fue la fase 5 (entrenamiento de modelos), en donde se experimentó con los 8 modelos de redes neuronales preeentrenadas con diferentes hiperparámetros.

El producto final de la fase 6 (clasificación de muestras) fue una tabla comparativa (Tabla III) con los valores correspondientes de exactitud para cada una de las diferentes arquitecturas evaluadas.

IV. Conjunto de datos

Las muestras utilizadas en este trabajo se obtuvieron de tres fuentes diferentes. El primer conjunto de datos contiene 14.972 muestras de malware, de las cuales 7.328 fueron obtenidas obtenidas de VirusShare [29] y 7.644 muestras fueron obtenidas del sitio VirusSign [30]. Por último 14.500 muestras de archivos benignos, pero con algún comportamiento similar al de los archivos maliciosos, fueron compartidas por la empresa ESET [31].

Estas muestras benignas fueron analizadas por un especialista humano porque presentaban algún comportamiento o

función similar a la del malware, por ejemplo, muchas de las muestras, por alguna razón, han sido cifradas o comprimidas utilizando algún *packer*; otras muestras cifran archivos o directorios, o acceden a funciones de cifrado; otras muestras chequean que no se esté ejecutando de forma virtualizada. Estos son algunos de los comportamientos o funciones que llamaron la atención de los especialistas humanos y que hacen que las muestras benignas tengan un comportamiento o alguna función similar a la del malware, sin llegar a serlo.

La Tabla I resume información sobre el conjunto de datos resultante de la combinación de los tres conjuntos de datos mencionados anteriormente.

TABLA I
CONJUNTO DE MUESTRAS DE MALWARE Y BENIGNAS

Tipo	Muestras	Fuente
Malware	7.328	VirusShare [29]
Malware	7.644	VirusSign [30]
Benigno	14.500	ESET [31]

Al momento de analizar las muestras de archivos benignos se percibió que muchas de las muestras eran identificadas como malware por otras empresas antivirus. Cuando se utiliza un método antivirus basado en firmas, algún comportamiento similar al del malware puede activar una detección falso positivo, por ejemplo el uso de *packers* o de detección de *Máquinas Virtuales* puede activar una firma antivirus.

Como queremos tener la mayor certeza posible de que las muestras benignas son en realidad benignas y suponiendo que los especialistas humanos pueden cometer errores de clasificación en algún momento, se optó por realizar una redefinición de archivo benigno.

Se determinó que una muestra es Benigna si menos del 3 % de los antivirus del sitio VirusTotal [32] detectan la muestra como "malware". Todas las muestras Benignas que son identificadas como malware por más del 3 % de los motores antivirus se excluyen del conjunto de datos. El valor de 3 % se decidió de forma arbitraria.

Los Datasets de muestras benignas que están disponibles en Internet, están constituidos por muestras de archivos que fueron obtenidos del computador, por ejemplo, se pueden encontrar Datasets con archivos del directorio "*Program Files (x86)*". Si bien son Datasets útiles la idea del presente trabajo es realizar experimentaciones con un Dataset de muestras benignas como fueron descriptas con anterioridad.

V. EXPERIMENTOS

Inicialmente se crearon 8 modelos de CNN para clasificar imágenes de archivos en dos clases: malware y benigno, la Tabla II muestra los 8 modelos utilizados con su respectiva arquitectura. Tal como se expresó en la Sección III, el experimento consta de seis fases, puede verse un modelo con las seis fases diagramadas en la Fig. 1.

Se utilizó una instancia de Google Colab en la que se configuraron los diferentes modelos. Las imágenes de archivos de muestra (malware y benigno) fueron almacenadas en una carpeta de Google Drive, donde las muestras se dividieron

TABLA II
MODELOS DE CNN EVALUADOS.

Modelo	Arquitectura	Modelo	Arquitectura
M1	ResNet-18	M5	VGG16
M2	ResNet-34	M6	DenseNet-121
M3	ResNet-50	M7	DenseNet-201
M4	ResNet-152	M8	AlexNet

aleatoriamente de la siguiente manera: el 70 % (20.630 muestras) para entrenamiento y el 30 % (8.842) para pruebas. Sólo hay dos categorías disponibles: malware y benigno.

La exactitud se utilizó como métrica principal para evaluar los modelos. Todos los modelos fueron entrenados con el mismo conjunto de datos, utilizando 50 épocas para cada modelo. Además se entrenaron con imágenes en color y en escala de grises de los siguientes tamaños: 64x64, 128x128, 224x224 y 256x256. En cuanto al tiempo total de entrenamiento de cada modelo (con descongelación - *unfreeze*), fue de aproximadamente 10 horas en promedio, utilizando un entorno con acelerador *Graphic Processing Unit* (GPU), no percibiendo ningún cambio notable al utilizar la aceleración *Tensor Processing Unit* (TPU) proporcionada por Colab.

Como todas las muestras obtenidas son de formato de archivos binarios Portable Ejecutable (PE) y nuestros modelos están entrenados para clasificar imágenes, debemos convertir dichas muestras en imágenes. En este trabajo todas las muestras se convirtieron a imágenes en escalas de grises en el rango [0,255] (0: negro, 255: blanco). El ancho de la imagen es fija y la altura puede variar en función del tamaño del archivo. Para las imágenes en color se crea un archivo RGB, de ancho fijo y altura variable, a partir de datos binarios de 24 bits: 8 bits rojos, 8 bits verdes y 8 bits azules. Para determinar el ancho de las imágenes se siguió el criterio utilizado en [33]. Además todas las imágenes fueron redimensionadas a 64x64, 128x128, 224x224 y 256x256 píxeles utilizando la interpolación bilineal. Ese redimensionamiento permite adaptar cada imagen a los datos de entrada de las arquitecturas CNN utilizadas en este trabajo.

Las redes neuronales convolucionales (CNN) son una clase de redes neuronales de aprendizaje profundo que se han utilizado con éxito en el reconocimiento de objetos a partir de imágenes. Las CNNs fueron introducidas por primera vez por Yann LeCun en 1998 [34]. En este trabajo se prueban ocho arquitecturas de CNN: ResNet-18; ResNet-34; ResNet-50; ResNet-152; VGG16; DenseNet-121; DenseNet-201 y AlexNet. Todas las CNN fueron seleccionadas, luego de una extensa revisión de la literatura para el problema, en función de su buen desempeño en tareas de clasificación de imágenes.

Desde su introducción, la arquitectura de red residual (ResNet) ha revolucionado el campo de la visión computacional y ha logrado establecer nuevos estándares en términos de exactitud y rendimiento. A pesar de su simplicidad, la ResNet ha demostrado ser extremadamente eficaz en una amplia variedad de tareas de visión computacional, incluyendo la clasificación, detección y localización de objetos [35]. La clave de su éxito radica en su arquitectura de "red residual", que permite que la red se entrene mejor, al permitir que la información fluya

a través de la red de forma más eficiente.

En los años posteriores con la aparición de las redes VGG16 y VGG19 [36], que son dos modelos de redes neuronales convolucionales muy populares entre los investigadores de visión computacional. Ambas fueron desarrolladas por el grupo de investigación Visual Geometry Group (VGG) de la Universidad de Oxford. VGG16 es el modelo usado en el ImageNet Large Scale Visual Recognition Challenge de 2014 (ILSVRC2014), donde obtuvo el primer lugar. En este trabajo utilizamos la arquitectura VGG-16 con normalización por lotes, debido a su simplicidad y robustez.

DenseNet es una red neuronal de tipo feed-forward que conecta todas las capas entre sí. La idea detrás de DenseNet es simplificar la propagación de la señal en la red al conectar todas las capas densamente. Esto se logra mediante el uso de una conexión skip, que permite que la señal se propague directamente de una capa a otra, sin pasar por las capas intermedias. Esto reduce el número de parámetros y mejora el flujo de la señal, lo que a su vez mejora el rendimiento de la red [37].

AlexNet es una arquitectura de red neuronal convolucional desarrollada por Alex Krizhevsky, Geoffrey Hinton e Ilya Sutskever [38]. AlexNet compitió en el ImageNet Large Scale Visual Recognition Challenge de 2012 (ILSVRC2012) [39] y ganó el primer lugar en la clasificación de objetos y el segundo en la localización de objetos. AlexNet fue la primera red neuronal en usar rectificadores en todas las capas ocultas, lo que permite un entrenamiento más rápido [40].

El objetivo principal de este trabajo es evaluar el desempeño de diferentes arquitecturas de CNN para clasificar Malware en imágenes de archivos ejecutables. Además encontramos la mejor tasa de identificación posible entre archivos malignos (malware) y archivos benignos. La Fig. 2 ilustra los pasos de la metodología adoptada.

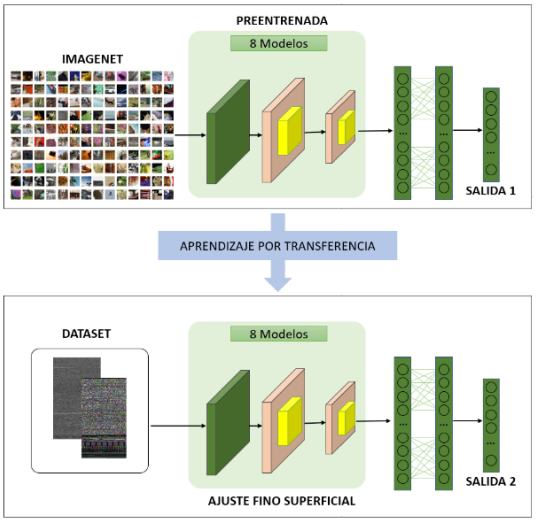

Fig. 2. Esquema de la arquitectura propuesta. Adaptación de Xiao *et al.* [41].

En primer lugar, seleccionamos una colección de imágenes de malware y de archivos benignos. A continuación, entrenamos una CNN con estas imágenes. Luego, medimos el desempeño de la CNN en términos de tasa de identificación de malware. Finalmente se analizaron los resultados.

El ajuste fino superficial o poco profundo [42] [43], *Shallow Fine-Tuning* (SFT), es un método de entrenamiento de redes neuronales que implica el ajuste de algunos de los parámetros de la red para mejorar el rendimiento en una tarea específica. Esto se contrasta con el entrenamiento profundo, en el que se ajustan todos los parámetros de la red. El SFT puede ser útil cuando se dispone de una red entrenada previamente que se puede reutilizar para una nueva tarea, y solo se requieren algunos ajustes para adaptarla a la nueva tarea.

Los modelos de CNN que se probaron a lo largo del presente trabajo se entrenaron utilizando el framework FastAI [44]. En lugar de construir y entrenar un modelo "desde cero" se utilizó el aprendizaje por transferencia para ahorrar tiempo y recursos al tener que entrenar varios modelos de aprendizaje automático desde cero para completar tareas similares. Siendo uno de los objetivos del presente trabajo.

Al tener un conjunto de datos balanceado (cantidades similares de muestras dentro de cada clase), como es el caso de este trabajo, disminuimos el sesgo de aprendizaje. El sesgo de aprendizaje ocurre cuando el algoritmo de aprendizaje prefiere una clase de datos sobre otra. Esto puede ocurrir si el conjunto de datos no está balanceado. Por ejemplo, si el conjunto de datos tiene más ejemplos de una clase que de otra, el algoritmo de aprendizaje puede sesgarse hacia la clase con más ejemplos.

Una métrica útil y utilizada cuando se tiene un conjunto de datos balanceado es la **exactitud** (escogida como métrica de evaluación del presente trabajo), del inglés "accuracy".

Además de la exactitud se utilizaron otras métricas como la ***precisión*** del inglés "precision", el ***recall*** y la ***f_beta***.

Uno de los mayores desafíos en el entrenamiento de las arquitecturas de CNN para tareas de clasificación es la poca diferencia de características significativas entre las muestras de malware y las muestras benignas, recordando que las muestras benignas de nuestro conjunto de datos, presentan alguna similitud respecto a las muestras de malware.

La Tabla III presenta el desempeño de cada arquitectura tomando como valor de comparación la exactitud. Los valores por debajo de los trabajos relacionados se deben a que la diferencia entre los archivos benignos y los archivos malware es muy superficial, los archivos benignos de nuestro dataset tienen alguna función similar a la de los malwares. En la tabla podemos apreciar que la arquitectura **DenseNet-201** obtuvo los mejores resultados para los tamaños de imágenes de: 224x224 y 256x256, cuyas precisiones fueron 96,33 % y 96,77 % respectivamente, sin denotar una diferencia importante para imágenes a color o en escala de grises. La otra arquitectura que obtuvo buenos resultados fue la **DenseNet-121** obteniendo 96,19 % (color) en el tamaño de imagen de 256x256.

En relación a las otras métricas evaluadas, en la Tabla IV podemos apreciar los valores correspondientes a la "Accuracy", "Precision", "Recall" y "F_beta" para la arquitectura DenseNet-201. Nuevamente podemos apreciar que los mejores resultados para estas métricas son obtenidos para los tamaños de imágenes mayores. Por una cuestión de espacios y extinción del presente trabajo solo se exponemos una arquitectura, la que denotó los mejores resultados.

TABLA III
TABLA COMPARATIVA DE LAS DIFERENTES ARQUITECTURAS CON SU CORRESPONDIENTE EXACTITUD

Arquitectura	64x64 Gris	64x64 Color	128x128 Gris	128x128 Color	224x224 Gris	224x224 Color	256x256 Gris	256x256 Color
ResNet-18	88,24 %	88,28 %	90,69 %	92,61 %	93,90 %	94,21 %	93,76 %	93,88 %
ResNet-34	89,52 %	88,76 %	91,82 %	91,64 %	94,42 %	94,56 %	94,09 %	94,63 %
ResNet-50	91,08 %	90,30 %	92,94 %	93,45 %	95,28 %	95,65 %	95,30 %	96,09 %
ResNet-152	91,64 %	91,17 %	94,34 %	93,97 %	95,61 %	95,78 %	95,86 %	95,86 %
VGG16	90,53 %	88,86 %	92,55 %	92,26 %	94,50 %	94,21 %	94,87 %	94,71 %
DenseNet-121	91,76 %	91,21 %	93,88 %	94,56 %	95,33 %	95,65 %	95,86 %	**96,19 %**
DenseNet-201	92,81 %	92,32 %	94,58 %	94,09 %	**96,17 %**	**96,33 %**	**96,73 %**	**96,77 %**
AlexNet	85,69 %	86,04 %	88,55 %	90,34 %	91,31 %	92,34 %	90,71 %	91,50 %

TABLA IV
RESULTADOS PARA LA ARQUITECTURA DENSENET-201

Tamaño	Carac.	Accuracy	Precision	Recall	F_beta
64x64	Gris	92,81 %	96,55 %	85,07 %	87,14 %
128x128	Gris	94,58 %	97,94 %	88,31 %	90,08 %
224x224	Gris	96,17 %	97,72 %	92,59 %	93,57 %
256x256	Gris	**96,73 %**	97,50 %	94,23 %	94,87 %
64x64	Color	92,32 %	**98,22 %**	82,29 %	85,05 %
128x128	Color	94,09 %	**98,03 %**	86,97 %	88,98 %
224x224	Color	**96,33 %**	**97,52 %**	93,20 %	94,04 %
256x256	Color	**96,77 %**	**97,30 %**	94,54 %	95,08 %

Si bien no se puede realizar una comparación directa de las exactitudes con los trabajos relacionados ya que se utilizan conjuntos de datos diferentes, podemos tomar como parámetro dichos valores. La diferencia de exactitud con respecto a los trabajo relacionados se debe, en gran parte, a que la mayoría de los trabajos analizados utiliza redes neuronales preentrenadas en combinación con otros métodos, volviéndolos más eficientes, mientras que otros trabajos utilizan redes neuronales creadas "desde cero" para abordar el problema.

Además como se expresó anteriormente al utilizar un dataset diferente al de los trabajos relacionados, y dado que las muestras benignas de este trabajo presentan alguna función similar a la del malware; la exactitud de las arquitecturas evaluadas tiene un rendimiento inferior en comparación a los trabajos investigados. Se puede apreciar que en comparación al trabajo realizado por Roseline *et al.* [12] que fue del 99,97 % hay una diferencia de 3,2 %; por otra parte al compararlo con el trabajo de Naeem *et al.* [28] cuya exactitud fue de 98,4 % podemos apreciar una diferencia de 1,63 %.

VI. CONCLUSIÓN

El presente trabajo busca optimizar la selección de muestras que un analista humano debería analizar utilizando para ello muestras de archivos benignos que tienen alguna funcionalidad similar a la de los malwares, en lugar de utilizar muestras de archivos benignos de programas conocidos o de archivos del propio computador, acortando con ello la brecha entre archivos malignos y archivos benignos, esto provoca una disminución de las diferencias visuales entre una imagen de malware y una imagen de archivo benigno.

En los experimentos realizados se puede apreciar que se alcanzó la mejor exactitud (**96,77 %**) con la arquitectura DenseNet-201 para imagen en color y tamaño de entrada de 256x256; la arquitectura DenseNet-121 también denoto exactitudes similares, 96,19 % para imágenes en color de tamaño 256x256. Estos datos experimentados muestran que es posible el usar redes neuronales preentrenadas y aprendizaje por transferencia para identificar muestras de malware.

Este trabajo contribuyó a crear un *benchmark* mediante una base de datos con archivos maliciosos y benignos representativos de la tarea que se modeló. Además se construyó una base de datos de archivos benignos con funciones similares a los archivos maliciosos, lo que permite acortar la brecha para la identificación. Por ultimo se ratificó la viabilidad sobre la utilización de redes neuronales preentrenadas y aprendizaje por transferencia para identificar archivos maliciosos utilizando un dataset casi homogéneo en lo que respecta a las funcionalidades de las muestras malignas y benignas.

Una particularidad que se puede observar en los experimentos, es que las redes con más capas ocultas (ResNet-152 y DenseNet-201) son las que arrojaron mejores resultados de exactitud, quedando para un trabajo futuro realizar pruebas con otras arquitecturas con mayor cantidad de capas ocultas. Además de las capas ocultas, se puede apreciar el tamaño de la imagen de entrada de 224x224 y 256x256 arroja los mejores resultados para todas las arquitecturas, quedando nuevamente para trabajos futuros experimentar con tamaños de imágenes mayores, por ejemplo, 512x512.

REFERENCIAS

[1] A. I. and K. A., "Android applications scanning: The guide," in *International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–5, Elsevier, 2019.
[2] H. J., R. S.A., G. S., K. S., and D. R., "An efficient densenet-based deep learning model for malware detection," *Entropy*, 2021.
[3] M. V. and A. A., "Malware detection based on code visualization and two-level classification," *Information*, 2021.
[4] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damaševičius, and T. Blažauskas, "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Applied Sciences*, vol. 10, no. 14, p. 4966, 2020.
[5] Y. Yang, X. Du, Z. Yang, and X. Liu, "Android malware detection based on structural features of the function call graph," *Electronics*, vol. 10, no. 2, p. 186, 2021.
[6] M. Nar, A. G. Kakisim, M. N. Yavuz, and I. Sogukpinar, "Analysis and comparison of disassemblers for opcode based malware analysis," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp. 17–22, 2019.
[7] S. Alsoghyer and I. Almomani, "Ransomware detection system for android applications," in *Electronics*, vol. 8, p. 868, CrossRef, 2019.
[8] H. Faris, M. Habib, I. Almomani, M. Eshtay, and I. Aljarah, "Optimizing extreme learning machines using chains of salps for efficient android ransomware detection," in *Applied Sciences*, vol. 10, p. 3706, CrossRef, 2020.

[9] J. Jeon, J. Kim, S. Jeon, S. Lee, and Y. Jeong, "Static analysis for malware detection with tensorflow and gpu," in *Advances in Computer Science and Ubiquitous Computing*, pp. 537–546, Springer, 2021.
[10] A. Mohaisen, O. Alrawi, and M. Mohaisen, "Amal: high-fidelity, behavior-based automated malware analysis and classification," *Computers & Security*, vol. 52, pp. 251–266, 2015.
[11] R. Sihwail, K. Omar, K. A. Zainol Ariffin, and S. Al Afghani, "Malware detection approach based on artifacts in memory image and dynamic analysis," *Applied Sciences*, vol. 9, no. 18, p. 3680, 2019.
[12] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," *IEEE Access*, vol. 8, pp. 206303–206324, 2020.
[13] V. Moussas and A. Andreatos, "Malware detection based on code visualization and two-level classification," in *Information*, vol. 12, p. 118, CrossRef, 2021.
[14] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damaševičius, and T. Blažauskas, "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Applied Sciences*, vol. 10, no. 14, p. 4966, 2020.
[15] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, 2019.
[16] S. Jang, S. Li, and Y. Sung, "Fasttext-based local feature visualization algorithm for merged image-based malware classification framework for cyber security and cyber defense," in *Mathematics*, vol. 8, p. 460, CrossRef, 2020.
[17] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, p. 107138, 2020.
[18] G. Xiao, J. Li, Y. Chen, and K. Li, "Malfcs: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks," *Journal of Parallel and Distributed Computing*, vol. 141, 04 2020.
[19] S. Saadat and V. Joseph Raymond, "Malware classification using cnn-xgboost model," in *Artificial Intelligence Techniques for Advanced Computing Applications*, pp. 191–202, Springer, 2021.
[20] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of cnn architectures (imcec)," *Computers & Security*, vol. 92, p. 101748, 2020.
[21] B. N. Narayanan and V. S. P. Davuluru, "Ensemble malware classification system using deep neural networks," *Electronics*, vol. 9, no. 5, p. 721, 2020.
[22] S. A. Roseline, A. Sasisri, S. Geetha, and C. Balasubramanian, "Towards efficient malware detection and classification using multilayered random forest ensemble technique," in *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6, IEEE, 2019.
[23] I. Ben Abdel Ouahab, M. Bouhorma, A. A. Boudhir, and L. El Aachak, "Classification of grayscale malware images using the k-nearest neighbor algorithm," in *The Proceedings of the Third International Conference on Smart City Applications*, pp. 1038–1050, Springer, 2019.
[24] H. Naeem, F. Ullah, M. R. Naeem, S. Khalid, D. Vasan, S. Jabbar, and S. Saeed, "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, p. 102154, 2020.
[25] M. Deore and U. Kulkarni, "Mdfrcnn: Malware detection using faster region proposals convolution neural network," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, no. 4, pp. 146–162, 2022.
[26] J. J. de Vicente Mohino, J. B. Higuera, J. R. B. Higuera, J. A. S. Montalvo, M. S. Rubio, and J. J. M. Herraiz, "Mmale—a methodology for malware analysis in linux environments," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 1447–1469, 2021.
[27] M. Dhalaria and E. Gandotra, "A hybrid approach for android malware detection and family classification," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 6, pp. 174–188, 2021.
[28] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas, and M. S. Javed, "Identification of malicious code variants based on image visualization," *Computers & Electrical Engineering*, vol. 76, pp. 225–237, 2019.
[29] VirusShare, "Virusshare.com - because sharing is caring." https://virusshare.com/, 2022. Accedido el 03/02/2022.
[30] VirusSign, "Virussign | malware research & data center, threat intelligence." https://www.virussign.com/, 2022. Accedido el 03/02/2022.
[31] ESET, "Soluciones antivirus y de seguridad de internet." https://www.eset.com/ar/, 2022. Accedido el 03/02/2022.
[32] VirusTotal, "Virustotal." https://www.virustotal.com/, 2022. Accedido el 03/02/2022.
[33] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, (New York, NY, USA), Association for Computing Machinery, 2011.
[34] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, pp. 21–28, 1988.
[35] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv*, 2016.
[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.
[37] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2752–2761, 2018.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.
[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.
[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
[41] T. Xiao, L. Liu, K. Li, W. Qin, S. Yu, and Z. Li, "Comparison of transferred deep neural networks in ultrasonic breast masses discrimination," *BioMed research international*, vol. 2018, 2018.
[42] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.
[43] M. Izadyyazdanabadi, E. Belykh, M. Mooney, N. Martirosyan, J. Eschbacher, P. Nakaji, M. C. Preul, and Y. Yang, "Convolutional neural networks: Ensemble modeling, fine-tuning and unsupervised semantic localization for neurosurgical cle images," *Journal of Visual Communication and Image Representation*, vol. 54, pp. 10–20, 2018.
[44] Fast.ai, "fast.ai - making neural nets uncool again." https://www.fast.ai/about/, 2022. Accedido el 03/02/2022.

Eduardo Malvacio Graduado de la Facultad de Ingeniería del Ejército Argentino (2013), especialista en guerra cibernética por el Centro de Comunicaciones y Guerra Electrónica del Ejército de Brasil (2014) y especialista en criptografía y seguridad informática por la Facultad de Ingeniería del Ejército Argentino (2016). Eduardo tiene experiencia en los siguientes temas: ciberseguridad y criptografía.

Julio Cesar Duarte Graduado del Instituto Militar de Ingeniería (1998), máster en Informática por la Pontificia Universidad Católica de Río de Janeiro (2003) y doctorado en Informática por la Pontificia Universidad Católica de Río de Janeiro (2009). Julio tiene experiencia multidisciplinaria, actuando en los siguientes temas: aprendizaje automático, inteligencia artificial y procesamiento del lenguaje natural.