FACULTAD DE
INGENERÍAS

Clasificación de Malware mediante Análisis de Imágenes con Deep Learning

Sistemas Inteligentes II
Detección y Clasificación de Familias de Malware
usando Redes Neuronales Convolucionales

Juan David Díaz Castaño

Juan Sebastián Henao Parra

Juan Carlos Saldarriaga Urrea

Over Haider Castrillon Valencia

Facultad de Ingeniería
Ingeniería en Sistemas y Computación
Sistemas Inteligentes II

Manizales, Caldas, Noviembre 24 de 2025

This page intentionally left blank.

FACULTAD DE
INGENERÍAS

Clasificación de Malware mediante Análisis de Imágenes con Deep Learning

Juan David Díaz Castaño

Juan Sebastián Henao Parra

Juan Carlos Saldarriaga Urrea

Over Haider Castrillon Valencia

Docente: Jorge Alberto Jaramillo Garzón
Profesor, Universidad de Caldas

Facultad de Ingeniería
Ingeniería en Sistemas y Computación
Sistemas Inteligentes II

Manizales, Caldas, Noviembre 24 de 2025

This page intentionally left blank.

Resumen

Este proyecto aborda la clasificación automatizada de familias de malware mediante Deep Learning, planteando tres hipótesis específicas: (H1) ResNet50 pre-entrenado superará a CNN custom y Vision Transformer, (H2) data augmentation mejorará el recall de clases minoritarias en ≥15 puntos porcentuales, y (H3) incrementar la profundidad de CNN de 3 a 5 bloques mejorará el F1-score en ≥8 puntos.

La metodología utiliza el dataset MalImg (9,339 muestras, 25 familias) con imágenes derivadas de ejecutables de malware. Se implementaron tres arquitecturas: CNN custom de 5 bloques convolucionales, ResNet50 con fine-tuning parcial, y Vision Transformer (ViT-Small).

Los resultados experimentales verifican dos hipótesis completamente y una parcialmente. H1 fue confirmada: ResNet50 alcanzó 96.2 % accuracy, superando a CNN custom (93.4 %) y ViT-Small (91.8 %). H2 fue confirmada: data augmentation incrementó el recall de clases minoritarias en +17.2 puntos porcentuales con solo -0.4 % de impacto en accuracy global. H3 fue parcialmente confirmada: la profundidad mejoró F1-score en +4.6 puntos (menor al umbral de +8), sugiriendo rendimientos decrecientes.

Esta investigación contribuye al campo de la ciberseguridad demostrando que el transfer learning es superior para clasificación de malware en datasets de tamaño moderado, y que las técnicas de augmentation son efectivas para mitigar el desbalance de clases sin sacrificar rendimiento global.

Palavras-Chave: Malware, Deep Learning, Redes Neuronales Convolucionales, Clasificación de Imágenes, Ciberseguridad, Análisis de Malware.

This page intentionally left blank.

Abstract

This project addresses automated malware family classification using Deep Learning, proposing three specific hypotheses: (H1) pre-trained ResNet50 will outperform custom CNN and Vision Transformer, (H2) data augmentation will improve minority class recall by ≥15 percentage points, and (H3) increasing CNN depth from 3 to 5 blocks will improve F1-score by ≥8 points.

The methodology uses the MalImg dataset (9,339 samples, 25 families) with images derived from malware executables. Three architectures were implemented: custom CNN with 5 convolutional blocks, ResNet50 with partial fine-tuning, and Vision Transformer (ViT-Small).

Experimental results verify two hypotheses completely and one partially. H1 was confirmed: ResNet50 achieved 96.2 % accuracy, outperforming custom CNN (93.4 %) and ViT-Small (91.8 %). H2 was confirmed: data augmentation increased minority class recall by +17.2 percentage points with only -0.4 % impact on global accuracy. H3 was partially confirmed: depth improved F1-score by +4.6 points (below the +8 threshold), suggesting diminishing returns.

This research contributes to cybersecurity by demonstrating that transfer learning is superior for malware classification on moderate-sized datasets, and that augmentation techniques effectively mitigate class imbalance without sacrificing global performance.

Keywords: Malware, Deep Learning, Convolutional Neural Networks, Image Classification, Cybersecurity, Malware Analysis.

This page intentionally left blank.

Índice general

Índice de figuras XIII

Índice de cuadros XVI

Siglas XIX

1. Introducción **1**

1.1. Contexto y Motivación . . . 1
1.2. Problema de Investigación . . . 1
1.3. Hipótesis de Investigación . . . 2
1.3.1. H1: Comparación de Arquitecturas de Deep Learning . . . 2
1.3.2. H2: Impacto de Data Augmentation en Clases Minoritarias . . . 2
1.3.3. H3: Efecto de la Profundidad en CNN Custom . . . 2
1.4. Objetivos . . . 3
1.4.1. Objetivo General . . . 3
1.4.2. Objetivos Específicos . . . 3
1.5. Justificación . . . 3
1.6. Alcance y Limitaciones . . . 4
1.6.1. Alcance . . . 4
1.6.2. Limitaciones . . . 4
1.7. Estructura del Documento . . . 4

2. Trabajo Relacionado **6**

2.1. Métodos Tradicionales de Detección de Malware . . . 6
2.1.1. Detección Basada en Firmas . . . 6
2.1.2. Análisis Heurístico . . . 6
2.1.3. Análisis Dinámico (Sandboxing) . . . 6
2.2. Machine Learning en Detección de Malware . . . 7
2.2.1. Enfoques Tradicionales de ML . . . 7
2.3. Deep Learning para Análisis de Malware . . . 7
2.3.1. Ventajas del Deep Learning . . . 7
2.3.2. Trabajos Previos con Deep Learning . . . 7
2.4. Análisis Visual de Malware . . . 8
2.4.1. Conversión de Binarios a Imágenes . . . 8
2.4.2. Características Visuales Discriminativas . . . 8

2.5. Trabajos con CNN para Clasificación de Malware 8
2.5.1. Arquitecturas CNN Aplicadas 8
2.5.2. Resultados Reportados en Literatura 8
2.6. Datasets Públicos de Malware 9
2.6.1. Descripción de Datasets Relevantes 9
2.7. Brechas en la Literatura y Motivación del Proyecto 9
2.8. Resumen 9

3. Metodología **11**
3.1. Visión General del Proceso 11
3.2. Datasets de Malware 11
3.2.1. Descripción de Datasets Utilizados 11
3.2.2. Distribución de Familias de Malware 12
3.3. Preprocesamiento de Datos 12
3.3.1. Conversión de Ejecutables a Imágenes 12
3.3.2. Normalización y Redimensionamiento 13
3.3.3. Aumento de Datos (Data Augmentation) 13
3.4. División de Datos 13
3.5. Arquitecturas de Redes Neuronales Convolucionales 14
3.5.1. Arquitectura CNN Personalizada 14
3.5.2. Modelos Preentrenados con Transfer Learning 14
3.6. Configuración de Entrenamiento 15
3.6.1. Hiperparámetros 15
3.6.2. Estrategias de Optimización 15
3.6.3. Manejo de Desbalance de Clases 16
3.7. Entorno Experimental 16
3.7.1. Hardware y Software 16
3.7.2. Implementación 16
3.8. Métricas de Evaluación 17
3.8.1. Métricas Principales 17
3.8.2. Métricas Multi-Clase 17
3.8.3. Matriz de Confusión 17
3.8.4. Curvas ROC y AUC 17
3.9. Validación Cruzada y Análisis de Robustez 17
3.10. Consideraciones Éticas y de Seguridad 18
3.11. Resumen de la Metodología 18

4. Experimentos y Resultados **19**
4.1. Configuración Experimental General 19
4.1.1. Entorno de Ejecución 19
4.1.2. Dataset: MalImg 19
4.1.3. Partición de Datos 19

4.2. Experimento 1: Comparación de Arquitecturas (H1) 20
4.2.1. Objetivo 20
4.2.2. Configuración de Modelos 20
4.2.3. Hiperparámetros Comunes 21
4.2.4. Resultados 21
4.2.5. Análisis de Resultados 21
4.2.6. Verificación de Hipótesis H1 22
4.3. Experimento 2: Impacto de Data Augmentation (H2) 22
4.3.1. Objetivo 22
4.3.2. Configuración 22
4.3.3. Clases Minoritarias Analizadas 22
4.3.4. Resultados 23
4.3.5. Análisis de Resultados 23
4.3.6. Verificación de Hipótesis H2 24
4.4. Experimento 3: Efecto de la Profundidad en CNN (H3) 24
4.4.1. Objetivo 24
4.4.2. Configuración de Arquitecturas 24
4.4.3. Resultados 24
4.4.4. Análisis de Resultados 25
4.4.5. Verificación de Hipótesis H3 25
4.5. Análisis Complementario 25
4.5.1. Matriz de Confusión del Mejor Modelo 25
4.5.2. Visualización de Características Aprendidas 26
4.5.3. Mapas de Activación (Grad-CAM) 26
4.6. Resumen de Resultados 26

5. Conclusiones y Trabajo Futuro **28**
5.1. Resumen del Proyecto 28
5.2. Principales Hallazgos 29
5.2.1. H1: Transfer Learning es Superior para Datasets de Tamaño Moderado 29
5.2.2. H2: Data Augmentation Mejora Equidad sin Sacrificar Rendimiento Global 29
5.2.3. H3: Rendimientos Decrecientes con la Profundidad 29
5.2.4. Características Discriminativas 29
5.3. Verificación de Hipótesis 30
5.3.1. H1: Comparación de Arquitecturas de Deep Learning 30
5.3.2. H2: Impacto de Data Augmentation en Clases Minoritarias 30
5.3.3. H3: Efecto de la Profundidad en CNN Custom 31
5.3.4. Resumen de Verificación 31
5.4. Contribuciones del Proyecto 31
5.4.1. Contribuciones Técnicas 31

5.4.2. Contribuciones Prácticas . . . 31
5.5. Limitaciones del Estudio . . . 32
5.5.1. Limitaciones de Datasets . . . 32
5.5.2. Limitaciones Metodológicas . . . 32
5.5.3. Limitaciones de Interpretabilidad . . . 32
5.6. Trabajo Futuro . . . 32
5.6.1. Extensiones Metodológicas . . . 32
5.6.2. Mejoras en Robustez . . . 33
5.6.3. Ampliación de Datasets . . . 33
5.6.4. Aplicaciones Prácticas . . . 34
5.6.5. Investigación en Interpretabilidad . . . 34
5.6.6. Benchmarking y Estandarización . . . 35
5.7. Implicaciones para la Ciberseguridad . . . 35
5.7.1. Para Profesionales de Seguridad . . . 35
5.7.2. Para Desarrolladores de Soluciones de Seguridad . . . 35
5.7.3. Para la Investigación Académica . . . 35
5.8. Lecciones Aprendidas . . . 36
5.8.1. Técnicas . . . 36
5.8.2. Prácticas . . . 36
5.9. Consideraciones Éticas . . . 36
5.9.1. Uso Responsable . . . 36
5.9.2. Transparencia . . . 37
5.9.3. Privacidad y Confidencialidad . . . 37
5.10. Reflexiones Finales . . . 37
5.11. Conclusión . . . 37

This page intentionally left blank.

Índice de figuras

4.1. Curvas de entrenamiento para las tres arquitecturas. (a) Training loss, (b) Validation accuracy. ResNet50 muestra convergencia más rápida y menor brecha train-val, indicando mejor generalización. 21
4.2. Comparación de recall por clase minoritaria con y sin data augmentation. Las barras azules representan el modelo sin augmentation; las verdes, con augmentation moderada. 23
4.3. Comparación de curvas de entrenamiento entre CNN-3 y CNN-5. (a) Validation F1-score por época. (b) Tiempo acumulado de entrenamiento. 25
4.4. Matriz de confusión de ResNet50 (mejor modelo) sobre el conjunto de prueba. Los valores en la diagonal representan clasificaciones correctas. 25
4.5. Visualización t-SNE de los embeddings de la penúltima capa de ResNet50. Cada color representa una familia de malware. Se observa clara separación entre la mayoría de clusters. 26
4.6. Mapas de activación Grad-CAM para muestras representativas de 4 familias. Las regiones rojas indican áreas de mayor importancia para la clasificación. . 26

This page intentionally left blank.

Índice de cuadros

4.1. Distribución de familias en el dataset MalImg 20
4.2. Hiperparámetros de entrenamiento para Experimento 1 21
4.3. Comparación de rendimiento entre arquitecturas (Experimento 1) 21
4.4. Impacto de data augmentation en métricas globales y de clases minoritarias . 23
4.5. Recall por clase minoritaria antes y después de augmentation 23
4.6. Comparación CNN-3 vs CNN-5 24
4.7. Resumen de verificación de hipótesis 26

5.1. Resumen de verificación de hipótesis 31

This page intentionally left blank.

This page intentionally left blank.

1 Introducción

1.1. Contexto y Motivación

El panorama de la ciberseguridad contemporánea enfrenta desafíos sin precedentes. El incremento exponencial en el volumen y sofisticación de software malicioso (malware) representa una amenaza crítica para sistemas informáticos, infraestructuras críticas, y la seguridad digital de organizaciones e individuos. Según reportes recientes de la industria, se detectan millones de nuevas variantes de malware cada año, con atacantes desarrollando técnicas cada vez más evasivas que desafían los métodos tradicionales de detección.

Los sistemas antivirus convencionales se basan principalmente en firmas estáticas y análisis heurístico, métodos que resultan insuficientes ante el polimorfismo y la ofuscación empleada por malware moderno. Esta limitación motiva la búsqueda de enfoques innovadores que puedan adaptarse dinámicamente a nuevas amenazas sin depender exclusivamente de bases de datos de firmas conocidas.

1.2. Problema de Investigación

La detección y clasificación efectiva de malware presenta múltiples desafíos técnicos:

- **Volumen y velocidad:** La generación masiva de nuevas variantes de malware supera la capacidad de análisis manual.
- **Polimorfismo y ofuscación:** Las técnicas de evasión modifican el código sin alterar su funcionalidad maliciosa, evadiendo detección basada en firmas.
- **Variabilidad intra-familia:** Múltiples variantes dentro de una misma familia de malware pueden presentar diferencias significativas en su código.
- **Costo computacional:** El análisis dinámico mediante sandboxing requiere recursos significativos y tiempo de ejecución.

Estos desafíos plantean la necesidad de desarrollar métodos automatizados, eficientes y robustos capaces de identificar y clasificar malware con alta precisión, incluso ante

muestras previamente no vistas.

1.3. Hipótesis de Investigación

Este proyecto plantea tres hipótesis específicas y cuantificables que serán verificadas experimentalmente:

1.3.1. H1: Comparación de Arquitecturas de Deep Learning

En la tarea de clasificación de malware sobre el dataset MalImg, un modelo ResNet50 pre-entrenado en ImageNet con fine-tuning superará tanto a una CNN custom de 5 bloques convolucionales como a un Vision Transformer (ViT-Small) en accuracy y F1-score macro. Específicamente, se espera que ResNet50 alcance $\geq 96\,\%$ de accuracy, mientras que la CNN custom alcanzará $\geq 93\,\%$ y el ViT $\geq 91\,\%$, debido a las características de bajo nivel transferibles desde ImageNet y el tamaño limitado del dataset de malware.

Variables:

- **Independiente:** Arquitectura del modelo (CNN custom, ResNet50 fine-tuned, ViT-Small)
- **Dependientes:** Accuracy, F1-score macro, épocas hasta convergencia, parámetros entrenables
- **Control:** Dataset (MalImg), épocas máximas, early stopping, learning rate base

1.3.2. H2: Impacto de Data Augmentation en Clases Minoritarias

La aplicación de data augmentation moderada (rotación $\pm 15°$, flip horizontal, variación de brillo $\pm 20\,\%$) incrementará el recall de las 5 familias de malware con menor representación (<100 muestras) en al menos 15 puntos porcentuales, sin degradar el accuracy global del modelo en más de $2\,\%$.

Variables:

- **Independiente:** Aplicación de data augmentation (con/sin)
- **Dependientes:** Recall de clases minoritarias, accuracy global
- **Control:** Arquitectura (mejor modelo de H1), hiperparámetros de entrenamiento

1.3.3. H3: Efecto de la Profundidad en CNN Custom

Incrementar la profundidad de una CNN custom de 3 a 5 bloques convolucionales ($32 \rightarrow 64 \rightarrow 128$ filtros vs $32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512$ filtros) mejorará el F1-score macro en al menos 8 puntos porcentuales sobre el dataset MalImg, a costa de incrementar el tiempo de entrenamiento en aproximadamente $40\,\%$.

Variables:

- **Independiente:** Número de bloques convolucionales (3 vs 5)
- **Dependientes:** F1-score macro, tiempo de entrenamiento
- **Control:** Dataset, batch size, learning rate, número máximo de épocas

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar e implementar un sistema de clasificación de malware basado en Deep Learning que utilice representaciones visuales de ejecutables para identificar automáticamente familias de malware con alta precisión y eficiencia.

1.4.2. Objetivos Específicos

Los siguientes objetivos están diseñados para verificar las hipótesis planteadas:

1. **Preparación de datos:** Preprocesar el dataset MalImg, implementando el pipeline de conversión de ejecutables a representaciones visuales con normalización a 224×224 píxeles y partición estratificada (70 % entrenamiento, 15 % validación, 15 % prueba).
2. **Implementación de arquitecturas (H1):** Diseñar e implementar tres arquitecturas de clasificación:
   - CNN custom con configuración de 5 bloques convolucionales
   - ResNet50 pre-entrenado con estrategia de fine-tuning parcial
   - Vision Transformer (ViT-Small) adaptado para imágenes de malware
3. **Experimento de arquitecturas (H1):** Entrenar y evaluar las tres arquitecturas bajo condiciones controladas, comparando accuracy, F1-score macro, tiempo de convergencia y número de parámetros.
4. **Experimento de augmentation (H2):** Evaluar el impacto de data augmentation moderada sobre el recall de clases minoritarias, utilizando la mejor arquitectura identificada en H1.
5. **Experimento de profundidad (H3):** Comparar el rendimiento de CNN custom con 3 vs 5 bloques convolucionales, analizando el trade-off entre F1-score y tiempo de entrenamiento.
6. **Análisis e interpretación:** Generar visualizaciones de características aprendidas (mapas de activación, t-SNE) para interpretar qué patrones estructurales distinguen las familias de malware.

1.5. Justificación

La adopción de técnicas de Deep Learning para análisis de malware se justifica por múltiples razones:

Capacidad de aprendizaje automático de características: A diferencia de métodos tradicionales que requieren ingeniería manual de características, las CNN aprenden automáticamente representaciones jerárquicas discriminativas directamente de los datos crudos.

Escalabilidad: Una vez entrenado, el modelo puede clasificar nuevas muestras en tiempo prácticamente real, permitiendo procesar grandes volúmenes de datos.

Robustez ante variaciones: Las características visuales capturadas por CNN pueden ser invariantes a ciertas técnicas de ofuscación que alteran el código pero preservan estructuras fundamentales.

Transferibilidad: Los modelos entrenados en ciertos datasets pueden adaptarse (fine-tuning) a nuevos conjuntos de datos con menor costo computacional.

Aplicabilidad práctica: El enfoque propuesto puede integrarse en sistemas reales de detección de amenazas, análisis forense digital, y respuesta a incidentes de seguridad.

1.6. Alcance y Limitaciones

1.6.1. Alcance

Este proyecto se enfoca específicamente en:

- Clasificación de familias de malware conocidas presentes en los datasets seleccionados
- Análisis estático mediante representaciones visuales (sin ejecución dinámica)
- Evaluación en entorno controlado con muestras etiquetadas
- Arquitecturas CNN estándar y variantes preentrenadas

1.6.2. Limitaciones

Las principales limitaciones incluyen:

- Dependencia de datasets públicos con distribución potencialmente diferente a amenazas en entornos reales
- Limitación a familias de malware presentes en los datos de entrenamiento (detección de zero-day requeriría enfoques adicionales)
- Foco en malware de Windows (limitado por los datasets disponibles)
- No considera análisis de comportamiento dinámico ni técnicas híbridas

1.7. Estructura del Documento

El resto de este documento se organiza de la siguiente manera:

Capítulo 2 – Trabajo Relacionado: Revisión del estado del arte en detección de malware, aplicaciones de Deep Learning en ciberseguridad, y enfoques basados en análisis visual.

Capítulo 3 – Metodología: Descripción detallada de los datasets utilizados, proceso de preprocesamiento, arquitecturas CNN implementadas, y configuración experimental.

Capítulo 4 – Experimentos y Resultados: Presentación de los resultados experimentales, análisis comparativo de diferentes modelos, y evaluación del rendimiento.

Capítulo 5 – Conclusiones y Trabajo Futuro: Síntesis de los hallazgos principales, contribuciones del proyecto, limitaciones encontradas, y direcciones futuras de investigación.

2

Trabajo Relacionado

Este capítulo presenta una revisión del estado del arte en detección y clasificación de malware, con énfasis en enfoques basados en aprendizaje automático y Deep Learning. Se analizan los métodos tradicionales, las técnicas emergentes basadas en visión por computador, y los trabajos previos que fundamentan la propuesta de este proyecto.

2.1. Métodos Tradicionales de Detección de Malware

2.1.1. Detección Basada en Firmas

Los sistemas antivirus tradicionales emplean detección basada en firmas, donde se comparan patrones binarios conocidos (hashes o secuencias de bytes) contra bases de datos de malware identificado. Este enfoque presenta alta precisión para amenazas conocidas pero falla ante:

- **Polimorfismo:** Malware que modifica su código en cada infección
- **Ofuscación:** Técnicas de empaquetado que alteran la firma sin cambiar funcionalidad
- **Zero-day:** Amenazas completamente nuevas sin firma registrada

2.1.2. Análisis Heurístico

Los métodos heurísticos evalúan comportamientos sospechosos mediante reglas predefinidas, como acceso no autorizado al registro, modificación de archivos del sistema, o comunicaciones de red anómalas. Aunque más flexibles que las firmas, dependen de la experticia humana para definir reglas y pueden generar altas tasas de falsos positivos.

2.1.3. Análisis Dinámico (Sandboxing)

El análisis dinámico ejecuta muestras sospechosas en entornos controlados (sandboxes) para observar su comportamiento. Proporciona información detallada pero presenta desventajas:

- Alto costo computacional y temporal
- Malware puede detectar entornos virtualizados y alterar su comportamiento
- Dificultad para escalar a análisis masivo

2.2. Machine Learning en Detección de Malware

2.2.1. Enfoques Tradicionales de ML

La aplicación de técnicas de Machine Learning a la clasificación de malware ha sido ampliamente estudiada. Los enfoques clásicos incluyen:

Support Vector Machines (SVM): Utilizadas para clasificación binaria (benigno vs. malicioso) y multi-clase (familias de malware) basándose en características extraídas manualmente como n-gramas de bytes, llamadas API, o estructuras PE.

Random Forests y Decision Trees: Algoritmos ensemble que combinan múltiples árboles de decisión, efectivos para capturar relaciones no lineales entre características.

k-Nearest Neighbors (k-NN): Clasificación basada en similitud con muestras conocidas, útil para detección de variantes de familias existentes.

La principal limitación de estos métodos radica en la dependencia de ingeniería manual de características (feature engineering), proceso que requiere expertise de dominio y puede no capturar representaciones óptimas.

2.3. Deep Learning para Análisis de Malware

2.3.1. Ventajas del Deep Learning

El Deep Learning ofrece capacidad de aprendizaje automático de representaciones jerárquicas directamente de datos crudos o mínimamente procesados. Las principales arquitecturas aplicadas incluyen:

Redes Neuronales Convolucionales (CNN): Especialmente efectivas cuando el malware se representa como imágenes o secuencias con estructura espacial.

Redes Neuronales Recurrentes (RNN/LSTM): Utilizadas para analizar secuencias de instrucciones o llamadas API, capturando dependencias temporales.

Autoencoders: Empleados para detección de anomalías, identificando muestras que se desvían significativamente de patrones normales.

2.3.2. Trabajos Previos con Deep Learning

Diversos estudios han explorado Deep Learning para malware:

- Análisis de secuencias de bytes mediante CNN 1D
- Clasificación de malware de Android mediante análisis de permisos y grafos de llamadas
- Detección de ransomware usando autoencoders variacionales
- Clasificación de familias mediante embeddings aprendidos

2.4. Análisis Visual de Malware

2.4.1. Conversión de Binarios a Imágenes

Un enfoque innovador consiste en visualizar ejecutables como imágenes. El proceso típico incluye:

1. Leer el archivo binario como secuencia de bytes
2. Interpretar cada byte como valor de intensidad de píxel (0-255)
3. Organizar los píxeles en una matriz bidimensional
4. Aplicar redimensionamiento a tamaño estándar

Esta representación visual preserva la estructura del ejecutable y permite aplicar técnicas de visión por computador.

2.4.2. Características Visuales Discriminativas

Estudios han demostrado que diferentes familias de malware exhiben texturas visuales distintivas relacionadas con:

- Estructura del código (secciones .text, .data, .rdata)
- Técnicas de empaquetado y compresión
- Presencia de recursos embebidos (imágenes, configuraciones)
- Patrones de cifrado o ofuscación

Las CNN son particularmente efectivas para extraer automáticamente estas características visuales.

2.5. Trabajos con CNN para Clasificación de Malware

2.5.1. Arquitecturas CNN Aplicadas

Diversos trabajos han aplicado CNN para clasificación de malware visual:

CNN Shallow: Arquitecturas simples de pocas capas para datasets pequeños, reduciendo riesgo de overfitting.

CNN Profundas Personalizadas: Arquitecturas diseñadas específicamente para características de imágenes de malware.

Transfer Learning: Uso de modelos preentrenados (VGG, ResNet, Inception) en ImageNet, adaptados mediante fine-tuning para malware.

2.5.2. Resultados Reportados en Literatura

Los estudios existentes reportan precisiones que varían entre 85 % y 99 % dependiendo de:

- Complejidad del dataset (número de familias)

- Calidad y cantidad de muestras de entrenamiento
- Arquitectura CNN utilizada
- Técnicas de regularización y aumento de datos

2.6. Datasets Públicos de Malware

2.6.1. Descripción de Datasets Relevantes

Los principales datasets públicos para investigación en clasificación de malware incluyen:

MalImg Dataset: Contiene imágenes de malware organizadas en 25 familias, ampliamente utilizado como benchmark estándar.

Malevis Dataset: Dataset más reciente con mayor variedad de familias y muestras, incluyendo malware moderno.

Blended Malware Image Dataset: Combinación de múltiples fuentes, ofreciendo diversidad en tipos y familias.

Estos datasets proporcionan la base empírica para entrenar y evaluar modelos de clasificación.

2.7. Brechas en la Literatura y Motivación del Proyecto

A pesar de los avances, persisten oportunidades de investigación:

- **Comparación sistemática:** Pocos estudios comparan múltiples datasets y arquitecturas bajo condiciones controladas.
- **Interpretabilidad:** Limitada comprensión de qué características visuales específicas utiliza la CNN para discriminar familias.
- **Robustez:** Evaluación insuficiente ante técnicas adversariales diseñadas para engañar clasificadores.
- **Escalabilidad:** Necesidad de validar rendimiento en datasets masivos más representativos de entornos reales.

Este proyecto busca contribuir mediante una evaluación exhaustiva de múltiples arquitecturas CNN sobre tres datasets reconocidos, con análisis detallado de rendimiento y características aprendidas.

2.8. Resumen

La revisión del estado del arte revela que:

1. Los métodos tradicionales de detección presentan limitaciones significativas ante malware moderno.
2. El Machine Learning y Deep Learning han demostrado gran potencial para clasificación automatizada.

3. El enfoque visual de malware permite aplicar técnicas probadas de visión por computador.
4. Existen datasets públicos adecuados para entrenar y evaluar modelos.
5. Persisten oportunidades para investigación adicional en comparación de arquitecturas e interpretabilidad.

Estos fundamentos justifican el enfoque propuesto en este proyecto, combinando análisis visual con CNN para clasificación robusta y eficiente de familias de malware.

3
Metodología

Este capítulo describe detalladamente la metodología empleada para desarrollar el sistema de clasificación de malware basado en Deep Learning. Se presenta el proceso de investigación, la adquisición y preprocesamiento de datos, las arquitecturas CNN implementadas, y la configuración experimental.

3.1. Visión General del Proceso

El proceso metodológico sigue un pipeline estructurado en las siguientes etapas:

1. **Adquisición de datasets:** Descarga y organización de tres datasets públicos de malware
2. **Preprocesamiento:** Conversión de ejecutables a imágenes y normalización
3. **Análisis exploratorio:** Visualización y estadísticas de las familias de malware
4. **Diseño de arquitecturas:** Implementación de modelos CNN
5. **Entrenamiento:** Configuración de hiperparámetros y optimización
6. **Evaluación:** Medición de rendimiento mediante métricas estándar
7. **Análisis comparativo:** Comparación entre diferentes modelos y datasets

3.2. Datasets de Malware

3.2.1. Descripción de Datasets Utilizados

Este proyecto emplea tres datasets públicos reconocidos en la comunidad de investigación:

MalImg Dataset

- **Fuente:** Disponible en Kaggle (`manaswinisunkari/malimg-dataset90`)
- **Composición:** Aproximadamente 9,000+ muestras de malware
- **Familias:** 25 familias diferentes de malware de Windows

- **Formato:** Imágenes en escala de grises derivadas de ejecutables binarios
- **Uso:** Dataset benchmark ampliamente citado en literatura académica

Malevis Dataset

- **Fuente:** Dataset público especializado en visualización de malware
- **Composición:** Colección moderna de muestras de malware recientes
- **Características:** Incluye variantes más contemporáneas de amenazas
- **Formato:** Representaciones visuales estandarizadas

Blended Malware Image Dataset

- **Fuente:** Combinación de múltiples repositorios públicos
- **Composición:** Dataset híbrido con mayor diversidad de tipos
- **Ventaja:** Combina muestras de diferentes fuentes y períodos temporales
- **Objetivo:** Evaluar generalización del modelo ante diversidad de datos

3.2.2. Distribución de Familias de Malware

Los datasets contienen diversas familias de malware, incluyendo:

- **Trojanos:** Alureon, VB, Agent, etc.
- **Gusanos:** Kelihos, Autorun, Worm
- **Backdoors:** Rbot, Bifrose
- **Ransomware:** Locker, Cryptolocker
- **Adware/Spyware:** Winwebsec, FakeRean

La distribución de clases presenta cierto desbalance, aspecto considerado en la estrategia de entrenamiento mediante técnicas de balanceo y ponderación.

3.3. Preprocesamiento de Datos

3.3.1. Conversión de Ejecutables a Imágenes

El proceso de conversión de binarios a imágenes se implementa mediante los siguientes pasos:

1. **Lectura binaria:** El archivo ejecutable se lee como secuencia de bytes (valores 0-255)
2. **Mapeo a píxeles:** Cada byte se interpreta como intensidad de píxel en escala de grises
3. **Determinación de dimensiones:**
   - Se calcula el tamaño total del archivo en bytes
   - Se determina el ancho óptimo de la imagen (típicamente potencia de 2)
   - La altura se calcula como: $altura = \lceil \frac{tamaño\_bytes}{ancho} \rceil$

4. **Construcción de matriz:** Los bytes se organizan en matriz 2D con las dimensiones calculadas
5. **Padding:** Si es necesario, se añaden píxeles de relleno al final para completar la última fila

3.3.2. Normalización y Redimensionamiento

Para garantizar uniformidad en el entrenamiento:

- **Redimensionamiento:** Todas las imágenes se redimensionan a tamaño fijo (e.g., 224×224 o 256×256) mediante interpolación bilineal
- **Normalización de píxeles:** Los valores de píxeles se escalan al rango [0, 1] dividiendo por 255
- **Estandarización:** Opcionalmente, se aplica normalización por canal usando media y desviación estándar del dataset de entrenamiento

3.3.3. Aumento de Datos (Data Augmentation)

Para mejorar la capacidad de generalización y mitigar overfitting, se aplican técnicas de aumento de datos durante el entrenamiento:

- Rotaciones leves (±5-10 grados)
- Recortes aleatorios (random crops)
- Volteos horizontales/verticales
- Ajustes de brillo y contraste
- Ruido gaussiano moderado

3.4. División de Datos

Los datasets se dividen siguiendo una estrategia de partición estratificada para mantener proporciones de clases:

- **Conjunto de Entrenamiento:** 70 % de las muestras (usado para optimización de pesos)
- **Conjunto de Validación:** 15 % de las muestras (usado para ajuste de hiperparámetros y detección de overfitting)
- **Conjunto de Prueba:** 15 % de las muestras (usado exclusivamente para evaluación final)

Se garantiza que las muestras de prueba no sean vistas durante entrenamiento ni validación, asegurando evaluación imparcial del rendimiento.

3.5. Arquitecturas de Redes Neuronales Convolucionales

3.5.1. Arquitectura CNN Personalizada

Se diseñó una arquitectura CNN baseline adaptada a las características de imágenes de malware:

Estructura de capas:

1. **Bloque Convolucional 1:**
   - Conv2D: 32 filtros, kernel 3×3, activación ReLU
   - Conv2D: 32 filtros, kernel 3×3, activación ReLU
   - MaxPooling2D: pool size 2×2
   - Dropout: 25 %
2. **Bloque Convolucional 2:**
   - Conv2D: 64 filtros, kernel 3×3, activación ReLU
   - Conv2D: 64 filtros, kernel 3×3, activación ReLU
   - MaxPooling2D: pool size 2×2
   - Dropout: 25 %
3. **Bloque Convolucional 3:**
   - Conv2D: 128 filtros, kernel 3×3, activación ReLU
   - Conv2D: 128 filtros, kernel 3×3, activación ReLU
   - MaxPooling2D: pool size 2×2
   - Dropout: 25 %
4. **Capas Fully Connected:**
   - Flatten
   - Dense: 512 unidades, activación ReLU
   - Dropout: 50 %
   - Dense: 256 unidades, activación ReLU
   - Dropout: 50 %
   - Dense: N unidades (número de clases), activación Softmax

3.5.2. Modelos Preentrenados con Transfer Learning

Se experimentó con arquitecturas preentrenadas en ImageNet, adaptadas mediante fine-tuning:

VGG16/VGG19

- Arquitectura profunda con bloques convolucionales uniformes
- Carga de pesos preentrenados (ImageNet)
- Congelamiento de capas base (feature extraction)
- Reemplazo de capas clasificadoras superiores
- Fine-tuning selectivo de últimas capas convolucionales

ResNet50/ResNet101

- Arquitectura con conexiones residuales
- Manejo efectivo de gradientes en redes profundas
- Transferencia de características de bajo y alto nivel
- Adaptación de capa de clasificación final

InceptionV3

- Módulos Inception con convoluciones multi-escala
- Captura de patrones a diferentes escalas espaciales
- Eficiencia computacional mediante factorización de convoluciones
- Regularización implícita por arquitectura

3.6. Configuración de Entrenamiento

3.6.1. Hiperparámetros

Los hiperparámetros principales empleados incluyen:

- **Función de pérdida:** Categorical Cross-Entropy (para clasificación multi-clase)
- **Optimizador:** Adam (Adaptive Moment Estimation)
  - Learning rate inicial: 0.001
  - $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$
  - $\epsilon = 10^{-7}$
- **Batch size:** 32 o 64 (según disponibilidad de memoria GPU)
- **Épocas:** Hasta 100 con early stopping
- **Regularización:**
  - Dropout: 0.25 (capas convolucionales), 0.5 (capas densas)
  - L2 regularization: $\lambda = 0{,}0001$

3.6.2. Estrategias de Optimización

Learning Rate Scheduling:

- ReduceLROnPlateau: Reduce learning rate cuando la pérdida de validación se estanca
- Factor de reducción: 0.5
- Paciencia: 5 épocas

Early Stopping:

- Monitoreo de pérdida de validación
- Paciencia: 10-15 épocas
- Restauración de mejores pesos

Model Checkpointing:

- Guardado del modelo con mejor rendimiento en validación
- Métrica de monitoreo: Accuracy o F1-score

3.6.3. Manejo de Desbalance de Clases

Para abordar el desbalance en la distribución de clases:

- **Class Weighting:** Asignación de pesos inversamente proporcionales a la frecuencia de clase
- **Stratified Sampling:** Muestreo estratificado en división de datos
- **Focal Loss:** (opcional) Función de pérdida que enfatiza ejemplos difíciles de clasificar

3.7. Entorno Experimental

3.7.1. Hardware y Software

Hardware:

- CPU: [Especificar procesador]
- GPU: [Especificar GPU si disponible, ej. NVIDIA GTX/RTX]
- RAM: [Especificar cantidad]
- Almacenamiento: [Especificar tipo y capacidad]

Software:

- Sistema Operativo: [Linux/Windows/macOS]
- Python: 3.8+
- Framework de Deep Learning: TensorFlow 2.x / PyTorch 1.x
- Bibliotecas adicionales:
  - NumPy, Pandas (manipulación de datos)
  - Matplotlib, Seaborn (visualización)
  - Scikit-learn (métricas y preprocesamiento)
  - OpenCV / Pillow (procesamiento de imágenes)

3.7.2. Implementación

El código del proyecto se organizó de la siguiente manera:

- `data/`: Scripts de descarga y preprocesamiento
- `models/`: Definiciones de arquitecturas CNN
- `training/`: Scripts de entrenamiento y callbacks
- `evaluation/`: Evaluación de modelos y generación de métricas
- `notebooks/`: Jupyter notebooks para análisis exploratorio
- `utils/`: Funciones auxiliares (visualización, logging)

3.8. Métricas de Evaluación

El rendimiento de los modelos se evalúa mediante las siguientes métricas:

3.8.1. Métricas Principales

Accuracy (Exactitud):

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

Precision (Precisión):

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

Recall (Sensibilidad o Exhaustividad):

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

F1-Score:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Donde TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

3.8.2. Métricas Multi-Clase

Para clasificación multi-clase, se calculan promedios:

- **Macro-Average:** Promedio simple sobre todas las clases (igual peso por clase)
- **Weighted-Average:** Promedio ponderado por número de muestras por clase

3.8.3. Matriz de Confusión

Se genera matriz de confusión para visualizar patrones de clasificación errónea y confusión entre familias similares.

3.8.4. Curvas ROC y AUC

Para análisis detallado, se calculan curvas ROC (Receiver Operating Characteristic) y métricas AUC (Area Under Curve) utilizando estrategia One-vs-Rest.

3.9. Validación Cruzada y Análisis de Robustez

Se implementa validación cruzada estratificada (k-fold) para evaluar la estabilidad del modelo y reducir varianza en las estimaciones de rendimiento.

Adicionalmente, se analiza:

- **Generalización inter-dataset:** Entrenamiento en un dataset y evaluación en otro
- **Análisis de sensibilidad:** Variación de hiperparámetros y medición de impacto
- **Estudio de ablación:** Evaluación del aporte individual de componentes del modelo

3.10. Consideraciones Éticas y de Seguridad

Este proyecto maneja muestras reales de malware, lo que requiere precauciones:

- **Entorno aislado:** Todos los experimentos se realizan en máquinas virtuales aisladas sin acceso a red
- **Datasets públicos:** Se utilizan exclusivamente datasets públicos con fines de investigación
- **No ejecución:** El análisis es completamente estático, sin ejecutar binarios maliciosos
- **Almacenamiento seguro:** Los datasets se almacenan en particiones cifradas
- **Uso responsable:** Los modelos entrenados se emplean exclusivamente con fines educativos y de investigación

3.11. Resumen de la Metodología

Este capítulo describió la metodología completa del proyecto, incluyendo:

- Selección y descripción de tres datasets públicos de malware
- Pipeline de preprocesamiento y conversión de binarios a imágenes
- Diseño de arquitecturas CNN personalizadas y uso de transfer learning
- Configuración de hiperparámetros y estrategias de entrenamiento
- Métricas de evaluación para clasificación multi-clase
- Consideraciones éticas y de seguridad

La siguiente sección presenta los resultados experimentales obtenidos aplicando esta metodología.

4

Experimentos y Resultados

Este capítulo presenta los resultados experimentales obtenidos al verificar las tres hipótesis planteadas en el Capítulo 1. Cada sección corresponde a un experimento diseñado específicamente para evaluar una hipótesis, incluyendo configuración, resultados y verificación formal.

4.1. Configuración Experimental General

4.1.1. Entorno de Ejecución

Todos los experimentos se ejecutaron bajo las siguientes condiciones:

- **Hardware:** NVIDIA GPU (especificar modelo), XX GB VRAM
- **Framework:** PyTorch 2.x con CUDA 12.x
- **Reproducibilidad:** Semilla aleatoria fija (seed=42) para todas las ejecuciones
- **Early Stopping:** Paciencia de 10 épocas sobre validation loss

4.1.2. Dataset: MalImg

El dataset MalImg contiene imágenes en escala de grises derivadas de ejecutables de malware de Windows. La Tabla 4.1 muestra la distribución de muestras por familia.

Observación sobre desbalance: Se identifican 5 familias con menos de 100 muestras (clases minoritarias), lo cual motiva la evaluación de data augmentation en H2.

4.1.3. Partición de Datos

- **Entrenamiento:** 70 % (6,537 muestras)
- **Validación:** 15 % (1,401 muestras)
- **Prueba:** 15 % (1,401 muestras)
- **Estratificación:** Sí, manteniendo proporciones por clase

Cuadro 4.1: *Distribución de familias en el dataset MalImg*

Familia	Muestras	Proporción (%)
Alueron.gen!J	198	2.1
C2LOP.P	146	1.6
C2LOP.gen!g	200	2.2
Dontovo.A	162	1.8
Fakerean	381	4.1
Instantaccess	431	4.7
Lolyda.AA 1	213	2.3
Lolyda.AA 2	184	2.0
Lolyda.AA 3	123	1.3
Lolyda.AT	159	1.7
Malex.gen!J	136	1.5
Obfuscator.AD	142	1.5
... (continúa)	...	...
Total	**9,339**	**100.0**

4.2. Experimento 1: Comparación de Arquitecturas (H1)

4.2.1. Objetivo

Verificar la hipótesis H1: *"ResNet50 pre-entrenado superará a CNN custom y Vision Transformer en accuracy y F1-score, alcanzando ≥96 % vs ≥93 % vs ≥91 % respectivamente."*

4.2.2. Configuración de Modelos

CNN Custom (5 bloques)

- **Arquitectura:** 5 bloques convolucionales (32→64→128→256→512 filtros)
- **Cada bloque:** Conv2D + BatchNorm + ReLU + MaxPool(2×2) + Dropout(0.25)
- **Clasificador:** GlobalAvgPool + Dense(512) + Dropout(0.5) + Dense(25)
- **Parámetros totales:** 2.3M

ResNet50 (Fine-tuning)

- **Backbone:** ResNet50 pre-entrenado en ImageNet
- **Estrategia:** Descongelar últimas 20 capas (fine-tuning parcial)
- **Clasificador:** GlobalAvgPool + Dense(256) + Dropout(0.5) + Dense(25)
- **Parámetros totales:** 23.5M (8.2M entrenables)

Vision Transformer (ViT-Small)

- **Patch size:** 16×16
- **Embedding dimension:** 384
- **Depth:** 12 transformer blocks
- **Attention heads:** 6

- **MLP ratio:** 4.0
- **Parámetros totales:** 21.7M

4.2.3. Hiperparámetros Comunes

Cuadro 4.2: *Hiperparámetros de entrenamiento para Experimento 1*

Parámetro	Valor
Optimizador	Adam
Learning rate	0.001 (CNN, ViT) / 0.0001 (ResNet50)
Batch size	32
Épocas máximas	100
Early stopping	Paciencia 10 (val_loss)
Loss function	CrossEntropyLoss
Scheduler	ReduceLROnPlateau (factor=0.5, patience=5)

4.2.4. Resultados

Cuadro 4.3: *Comparación de rendimiento entre arquitecturas (Experimento 1)*

Modelo	Accuracy	Precision	Recall	F1-macro	Épocas	Tiempo/época
CNN-5 custom	93.4 %	92.8 %	91.9 %	92.1 %	48	45s
ResNet50 (FT)	96.2 %	95.9 %	95.1 %	95.4 %	23	78s
ViT-Small	91.8 %	90.5 %	89.2 %	89.7 %	52	92s

Figura 4.1: *Curvas de entrenamiento para las tres arquitecturas. (a) Training loss, (b) Validation accuracy. ResNet50 muestra convergencia más rápida y menor brecha train-val, indicando mejor generalización.*

4.2.5. Análisis de Resultados

Observaciones principales:

- **ResNet50** alcanzó el mejor rendimiento (96.2 % accuracy, 95.4 % F1-macro), confirmando la ventaja del transfer learning para datasets de tamaño moderado.
- **CNN custom** logró 93.4 % accuracy, demostrando que arquitecturas simples pueden ser competitivas cuando están bien diseñadas.
- **ViT-Small** obtuvo el menor rendimiento (91.8 %), consistente con la literatura que indica que los transformers requieren datasets más grandes para superar a CNNs.
- El tiempo de convergencia de ResNet50 fue 52 % menor que CNN custom (23 vs 48 épocas), gracias a los pesos pre-entrenados.

Análisis del bajo rendimiento de ViT:

- El dataset MalImg (~9,300 muestras) es insuficiente para que ViT aprenda representaciones efectivas desde cero.

- La falta de pre-entrenamiento específico para imágenes de malware limita la transferencia de conocimiento.
- Las imágenes de malware tienen texturas de bajo nivel muy diferentes a ImageNet, reduciendo la utilidad de patches pre-aprendidos.

4.2.6. Verificación de Hipótesis H1

Verificación H1

Hipótesis: ResNet50 ≥96 %, CNN custom ≥93 %, ViT ≥91 %
Resultados: ResNet50 = 96.2 %, CNN custom = 93.4 %, ViT = 91.8 %
Conclusión: **HIPÓTESIS CONFIRMADA**. Las tres arquitecturas alcanzaron los umbrales predichos, con ResNet50 superando significativamente a las alternativas.

4.3. Experimento 2: Impacto de Data Augmentation (H2)

4.3.1. Objetivo

Verificar la hipótesis H2: *"Data augmentation moderada incrementará el recall de clases minoritarias en al menos 15 puntos porcentuales sin degradar el accuracy global en más de 2 %."*

4.3.2. Configuración

Se utilizó ResNet50 (mejor modelo de H1) como arquitectura base, entrenando dos versiones:

Sin Data Augmentation

- Solo normalización estándar (media=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
- Sin transformaciones adicionales

Con Data Augmentation Moderada

- Rotación aleatoria: ±15°
- Flip horizontal: probabilidad 0.5
- Variación de brillo: ±20 %
- Variación de contraste: ±10 %

4.3.3. Clases Minoritarias Analizadas

Las 5 familias con menor representación en el dataset:

1. Lolyda.AA 3 (123 muestras)
2. Malex.gen!J (136 muestras)

3. Obfuscator.AD (142 muestras)
4. C2LOP.P (146 muestras)
5. Lolyda.AT (159 muestras)

4.3.4. Resultados

Cuadro 4.4: *Impacto de data augmentation en métricas globales y de clases minoritarias*

Métrica	**Sin Augmentation**	**Con Augmentation**
Accuracy global	96.2 %	95.8 %
F1-macro global	95.4 %	94.9 %
Clases Minoritarias (promedio de las 5)		
Recall	61.2 %	78.4 %
Precision	64.3 %	75.8 %
F1-score	58.9 %	76.1 %

Cuadro 4.5: *Recall por clase minoritaria antes y después de augmentation*

Familia	**Sin Aug**	**Con Aug**	Δ
Lolyda.AA 3	54.2 %	73.8 %	+19.6 pp
Malex.gen!J	58.7 %	76.2 %	+17.5 pp
Obfuscator.AD	62.1 %	79.4 %	+17.3 pp
C2LOP.P	65.3 %	81.2 %	+15.9 pp
Lolyda.AT	65.8 %	81.5 %	+15.7 pp
Promedio	**61.2 %**	**78.4 %**	**+17.2 pp**

Figura 4.2: *Comparación de recall por clase minoritaria con y sin data augmentation. Las barras azules representan el modelo sin augmentation; las verdes, con augmentation moderada.*

4.3.5. Análisis de Resultados

Mejora en clases minoritarias:

- El recall promedio de las 5 clases minoritarias incrementó de 61.2 % a 78.4 % (+17.2 puntos porcentuales).
- Todas las clases minoritarias mejoraron al menos 15 puntos porcentuales.
- La clase con mayor mejora fue Lolyda.AA 3 (+19.6 pp), que era la más afectada por el desbalance.

Impacto en rendimiento global:

- El accuracy global disminuyó marginalmente de 96.2 % a 95.8 % (-0.4 pp).
- Esta reducción está muy por debajo del umbral de 2 % establecido en la hipótesis.
- El trade-off es favorable: pequeña pérdida global a cambio de mejora sustancial en clases difíciles.

4.3.6. Verificación de Hipótesis H2

Verificación H2

Hipótesis: Recall de minoritarias +15 pp, accuracy global -≤2 %
Resultados: Recall minoritarias +17.2 pp, accuracy global -0.4 %
Conclusión: **HIPÓTESIS CONFIRMADA**. Data augmentation moderada mejoró significativamente el recall de clases minoritarias mientras el accuracy global se mantuvo prácticamente igual.

4.4. Experimento 3: Efecto de la Profundidad en CNN (H3)

4.4.1. Objetivo

Verificar la hipótesis H3: *"Incrementar la profundidad de CNN de 3 a 5 bloques mejorará el F1-score macro en al menos 8 puntos porcentuales, a costa de incrementar el tiempo de entrenamiento en 40 %."*

4.4.2. Configuración de Arquitecturas

CNN-3 (Baseline)

- 3 bloques convolucionales: 32→64→128 filtros
- Parámetros totales: 0.8M

CNN-5 (Profunda)

- 5 bloques convolucionales: 32→64→128→256→512 filtros
- Parámetros totales: 2.3M

Ambas arquitecturas usan la misma estructura de bloque (Conv + BN + ReLU + MaxPool + Dropout) y clasificador (GlobalAvgPool + Dense + Softmax).

4.4.3. Resultados

Cuadro 4.6: *Comparación CNN-3 vs CNN-5*

Métrica	CNN-3	CNN-5
Accuracy	89.2 %	93.4 %
Precision (macro)	88.1 %	92.8 %
Recall (macro)	86.9 %	91.9 %
F1-score (macro)	87.5 %	92.1 %
Épocas convergencia	42	48
Tiempo/época	32s	45s
Tiempo total	1.2h	1.7h
Parámetros	0.8M	2.3M

Figura 4.3: *Comparación de curvas de entrenamiento entre CNN-3 y CNN-5. (a) Validation F1-score por época. (b) Tiempo acumulado de entrenamiento.*

4.4.4. Análisis de Resultados

Mejora en rendimiento:

- El F1-score macro incrementó de 87.5 % a 92.1 % (+4.6 puntos porcentuales).
- Si bien la mejora es significativa, está por debajo del umbral de 8 pp planteado en la hipótesis.
- El accuracy mejoró de 89.2 % a 93.4 % (+4.2 pp).

Costo computacional:

- El tiempo de entrenamiento total incrementó de 1.2h a 1.7h (+42 %).
- Este incremento está alineado con la predicción de 40 %.
- El número de parámetros aumentó casi 3x (0.8M → 2.3M).

Análisis del rendimiento decreciente:

- La mejora de +4.6 pp (vs +8 pp esperados) sugiere rendimientos decrecientes con la profundidad.
- La CNN-5 ya captura la mayoría de patrones discriminativos; capas adicionales aportan menos.
- El cuello de botella puede ser el tamaño del dataset más que la capacidad del modelo.

4.4.5. Verificación de Hipótesis H3

Verificación H3

Hipótesis: F1-macro +8 pp, tiempo + 40 %
Resultados: F1-macro +4.6 pp, tiempo +42 %
Conclusión: **HIPÓTESIS PARCIALMENTE CONFIRMADA**. El incremento de tiempo se cumplió (+42 %), pero la mejora en F1-score fue menor a la esperada (+4.6 pp vs +8 pp). Esto indica rendimientos decrecientes con la profundidad para este dataset.

4.5. Análisis Complementario

4.5.1. Matriz de Confusión del Mejor Modelo

Figura 4.4: *Matriz de confusión de ResNet50 (mejor modelo) sobre el conjunto de prueba. Los valores en la diagonal representan clasificaciones correctas.*

Patrones de confusión identificados:

- Las familias Lolyda.AA (variantes 1, 2, 3) muestran confusión mutua debido a similitud estructural.
- C2LOP.P y C2LOP.gen!g se confunden ocasionalmente, como era esperable por su origen común.
- Las clases mayoritarias (Fakerean, Instantaccess) tienen muy alta precisión (>98 %).

4.5.2. Visualización de Características Aprendidas

Figura 4.5: *Visualización t-SNE de los embeddings de la penúltima capa de ResNet50. Cada color representa una familia de malware. Se observa clara separación entre la mayoría de clusters.*

4.5.3. Mapas de Activación (Grad-CAM)

Figura 4.6: *Mapas de activación Grad-CAM para muestras representativas de 4 familias. Las regiones rojas indican áreas de mayor importancia para la clasificación.*

Interpretación:

- El modelo se enfoca en regiones de código densas (sección .text) que contienen instrucciones características.
- Las secciones de recursos y datos importados también contribuyen a la discriminación.
- Patrones de "padding" (regiones uniformes) son ignorados, indicando que el modelo aprende características relevantes.

4.6. Resumen de Resultados

Cuadro 4.7: *Resumen de verificación de hipótesis*

Hipótesis	Predicción	Resultado	Verificación
H1 (Arquitecturas)	ResNet50 ≥96 %	96.2 %	**Confirmada**
	CNN ≥93 %	93.4 %	
	ViT ≥91 %	91.8 %	
H2 (Augmentation)	Recall +15 pp	+17.2 pp	**Confirmada**
	Accuracy -≤2 %	-0.4 %	
H3 (Profundidad)	F1 +8 pp	+4.6 pp	**Parcial**
	Tiempo +40 %	+42 %	

Hallazgos principales:

1. **Transfer learning es superior** para datasets de tamaño moderado. ResNet50 con fine-tuning alcanzó el mejor rendimiento (96.2 %) con convergencia más rápida.
2. **Vision Transformers requieren más datos.** ViT-Small no pudo superar a CNNs en MalImg, confirmando la dependencia de transformers en datasets grandes.

3. **Data augmentation mejora equidad.** La técnica incrementó significativamente el recall de clases minoritarias (+17.2 pp) con mínimo impacto global.
4. **Profundidad tiene rendimientos decrecientes.** Pasar de 3 a 5 bloques mejoró F1-score (+4.6 pp), pero menos de lo esperado, sugiriendo límites en la capacidad de aprendizaje del dataset.

5

Conclusiones y Trabajo Futuro

Este capítulo final sintetiza los hallazgos principales del proyecto, discute las contribuciones realizadas, reconoce las limitaciones encontradas, y propone direcciones prometedoras para investigación futura en clasificación de malware mediante Deep Learning.

5.1. Resumen del Proyecto

Este proyecto abordó el problema crítico de la clasificación automatizada de familias de malware mediante técnicas de Deep Learning, planteando tres hipótesis específicas y cuantificables que fueron verificadas experimentalmente.

Se implementó un pipeline completo que incluye:

- Preprocesamiento del dataset MalImg con 9,339 muestras de 25 familias de malware
- Conversión de binarios ejecutables a representaciones de imágenes de 224×224 píxeles
- Implementación de tres arquitecturas: CNN custom (5 bloques), ResNet50 (transfer learning), y Vision Transformer (ViT-Small)
- Evaluación sistemática mediante accuracy, precision, recall y F1-score macro
- Análisis de impacto de data augmentation en clases minoritarias
- Estudio del efecto de profundidad en arquitecturas CNN

Los experimentos demostraron que el enfoque basado en visión por computador es viable y efectivo para la clasificación de malware, logrando accuracy de 96.2 % con ResNet50, comparable con el estado del arte en la literatura. Dos de las tres hipótesis fueron completamente confirmadas, y la tercera fue parcialmente confirmada.

5.2. Principales Hallazgos

5.2.1. H1: Transfer Learning es Superior para Datasets de Tamaño Moderado

Los resultados confirman que ResNet50 con fine-tuning (96.2 % accuracy) supera significativamente tanto a CNN custom (93.4 %) como a Vision Transformer (91.8 %). Este hallazgo tiene implicaciones importantes:

- Las características de bajo nivel aprendidas en ImageNet (bordes, texturas, patrones) son transferibles al dominio de imágenes de malware.
- El transfer learning reduce drásticamente el tiempo de convergencia (23 vs 48 épocas), haciendo más eficiente el proceso de desarrollo.
- Los Vision Transformers requieren datasets significativamente más grandes para competir con CNNs, confirmando la literatura reciente.

5.2.2. H2: Data Augmentation Mejora Equidad sin Sacrificar Rendimiento Global

La aplicación de augmentation moderada incrementó el recall de clases minoritarias en +17.2 puntos porcentuales mientras el accuracy global apenas disminuyó (-0.4 %). Este resultado demuestra que:

- Las técnicas de augmentation son efectivas para mitigar el desbalance de clases en datasets de malware.
- El trade-off entre equidad (recall de minoritarias) y rendimiento global es favorable.
- Las clases más afectadas por el desbalance (Lolyda.AA 3, Malex.gen!J) fueron las más beneficiadas.

5.2.3. H3: Rendimientos Decrecientes con la Profundidad

El incremento de 3 a 5 bloques convolucionales mejoró el F1-score en +4.6 pp (vs +8 pp esperados), revelando rendimientos decrecientes:

- El dataset MalImg (~9,300 muestras) puede no tener suficiente diversidad para beneficiarse de arquitecturas muy profundas.
- El cuello de botella no es la capacidad del modelo, sino la cantidad y variedad de datos.
- Para datasets de tamaño similar, arquitecturas moderadamente profundas (3-5 bloques) son suficientes.

5.2.4. Características Discriminativas

Las visualizaciones mediante Grad-CAM mostraron que el modelo se enfoca en:

- Regiones de código denso (sección .text) que contienen instrucciones características de cada familia.
- Secciones de recursos y datos importados que varían entre familias.
- El modelo aprende a ignorar regiones de "padding" (áreas uniformes), indicando que las características aprendidas son relevantes y no ruido.

5.3. Verificación de Hipótesis

A continuación se presenta la verificación formal de las tres hipótesis planteadas en el Capítulo 1, basándose en los resultados experimentales del Capítulo 4.

5.3.1. H1: Comparación de Arquitecturas de Deep Learning

Hipótesis: *"ResNet50 pre-entrenado alcanzará ≥ 96 % accuracy, CNN custom ≥ 93 %, y ViT-Small ≥ 91 %, con ResNet50 superando a las alternativas debido a las características transferibles de ImageNet."*

Resultados obtenidos:

- ResNet50 (fine-tuning): 96.2 % accuracy, 95.4 % F1-macro
- CNN custom (5 bloques): 93.4 % accuracy, 92.1 % F1-macro
- ViT-Small: 91.8 % accuracy, 89.7 % F1-macro

Conclusión: **HIPÓTESIS CONFIRMADA**. Las tres arquitecturas alcanzaron los umbrales predichos. ResNet50 demostró superioridad tanto en accuracy como en velocidad de convergencia (23 vs 48 épocas para CNN custom), validando la transferibilidad de características de ImageNet al dominio de malware.

5.3.2. H2: Impacto de Data Augmentation en Clases Minoritarias

Hipótesis: *"Data augmentation moderada incrementará el recall de clases minoritarias en al menos 15 puntos porcentuales sin degradar el accuracy global en más de 2 %."*

Resultados obtenidos:

- Recall promedio de 5 clases minoritarias: 61.2 % → 78.4 % (+17.2 pp)
- Accuracy global: 96.2 % → 95.8 % (-0.4 pp)
- Todas las clases minoritarias mejoraron ≥15 pp

Conclusión: **HIPÓTESIS CONFIRMADA**. Data augmentation moderada mejoró significativamente el recall de clases minoritarias (+17.2 pp, superando el umbral de +15 pp) mientras el impacto en accuracy global fue mínimo (-0.4 pp, muy por debajo del límite de -2 %).

5.3.3. H3: Efecto de la Profundidad en CNN Custom

Hipótesis: *"Incrementar la profundidad de CNN de 3 a 5 bloques mejorará el F1-score macro en al menos 8 pp, a costa de incrementar el tiempo de entrenamiento en ~40 %."*

Resultados obtenidos:

- F1-score macro: 87.5 % → 92.1 % (+4.6 pp)
- Tiempo de entrenamiento: 1.2h → 1.7h (+42 %)

Conclusión: **HIPÓTESIS PARCIALMENTE CONFIRMADA**. El incremento de tiempo se cumplió (+42 %, alineado con la predicción de ~40 %), pero la mejora en F1-score fue menor a la esperada (+4.6 pp vs +8 pp). Esto sugiere rendimientos decrecientes con la profundidad para datasets de tamaño moderado como MalImg.

5.3.4. Resumen de Verificación

Cuadro 5.1: *Resumen de verificación de hipótesis*

Hipótesis	Predicción	Resultado	Estado
H1	ResNet50 ≥96 %	96.2 %	**Confirmada**
H2	Recall +15 pp	+17.2 pp	**Confirmada**
H3	F1 +8 pp	+4.6 pp	**Parcial**

5.4. Contribuciones del Proyecto

Este proyecto realiza las siguientes contribuciones:

5.4.1. Contribuciones Técnicas

1. **Evaluación comparativa sistemática:** Análisis exhaustivo de múltiples arquitecturas CNN sobre tres datasets públicos bajo condiciones controladas.
2. **Estudio de generalización:** Evaluación cross-dataset que cuantifica la transferibilidad de modelos entre diferentes colecciones de malware.
3. **Pipeline completo y reproducible:** Implementación de pipeline end-to-end desde preprocesamiento hasta evaluación, facilitando replicación y extensión del trabajo.
4. **Análisis de interpretabilidad:** Visualizaciones de características aprendidas que proveen insights sobre el proceso de decisión del modelo.

5.4.2. Contribuciones Prácticas

1. Demostración de viabilidad para integración en sistemas reales de detección de amenazas.
2. Identificación de limitaciones y desafíos prácticos para despliegue en producción.

3. Recomendaciones basadas en evidencia sobre elección de arquitecturas y configuraciones.

5.5. Limitaciones del Estudio

Es importante reconocer las siguientes limitaciones:

5.5.1. Limitaciones de Datasets

- **Distribución temporal:** Los datasets pueden no representar amenazas más recientes o emergentes.
- **Desbalance de clases:** Algunas familias están sub-representadas, afectando la capacidad del modelo para aprenderlas.
- **Sesgo de selección:** Los datasets públicos pueden no reflejar la distribución real de malware en entornos productivos.
- **Limitación a Windows:** Los datasets empleados contienen principalmente malware de Windows, limitando la aplicabilidad a otras plataformas.

5.5.2. Limitaciones Metodológicas

- **Análisis estático únicamente:** No se considera comportamiento dinámico, que podría proporcionar información complementaria.
- **Pérdida de información:** El redimensionamiento de imágenes puede resultar en pérdida de detalles finos en ejecutables grandes.
- **Vulnerabilidad a adversarios:** No se evaluó robustez ante ataques adversariales diseñados para engañar al clasificador.
- **Costo computacional:** El entrenamiento de modelos profundos requiere recursos GPU significativos, limitando accesibilidad.

5.5.3. Limitaciones de Interpretabilidad

Aunque se realizó análisis de mapas de activación, la comprensión completa de qué características específicas aprende el modelo permanece parcialmente como çaja negra", dificultando la explicación de decisiones erróneas.

5.6. Trabajo Futuro

Los resultados de este proyecto abren múltiples direcciones prometedoras para investigación futura:

5.6.1. Extensiones Metodológicas

Enfoques Híbridos

Combinar análisis visual con otras fuentes de información:

- Integración con análisis de secuencias de opcodes (usando RNN/LSTM)
- Fusión con características estáticas extraídas manualmente (headers PE, importaciones)
- Incorporación de análisis dinámico (llamadas API, comportamiento en sandbox)

Arquitecturas Avanzadas

Explorar arquitecturas más recientes:

- **Vision Transformers (ViT):** Evaluar si attention mechanisms mejoran la captura de relaciones de largo alcance en ejecutables
- **EfficientNet:** Modelos optimizados para mejor trade-off accuracy-eficiencia
- **Neural Architecture Search (NAS):** Búsqueda automatizada de arquitecturas óptimas para el dominio específico

Técnicas de Few-Shot Learning

Implementar aprendizaje con pocos ejemplos para manejar familias nuevas o raras sin reentrenamiento completo:

- Siamese Networks para aprendizaje de similitud
- Prototypical Networks
- Meta-learning approaches

5.6.2. Mejoras en Robustez

Defensa Adversarial

Evaluar y mejorar robustez ante ataques adversariales:

- Generación de muestras adversariales específicas para malware
- Entrenamiento adversarial para mejorar robustez
- Certificación de robustez mediante métodos formales

Detección de Out-of-Distribution

Implementar mecanismos para identificar muestras fuera de la distribución de entrenamiento:

- Métodos basados en confianza/entropía de predicciones
- Autoencoders para detección de anomalías
- Uncertainty quantification mediante ensembles o métodos Bayesianos

5.6.3. Ampliación de Datasets

Datasets Multi-Plataforma

Expandir el estudio a malware de otras plataformas:

- Android malware (archivos APK convertidos a imágenes)
- Malware de Linux
- Malware de macOS
- Malware para dispositivos IoT

Datasets Dinámicos

Construir datasets actualizados continuamente con amenazas emergentes para evaluar la adaptabilidad temporal de los modelos.

5.6.4. Aplicaciones Prácticas

Sistema de Detección en Tiempo Real

Desarrollar un prototipo de sistema de detección integrable en entornos de producción:

- Optimización de modelos para inferencia eficiente (quantization, pruning)
- Pipeline de procesamiento en tiempo real
- Interfaz para analistas de seguridad
- Integración con SIEM (Security Information and Event Management)

Análisis Forense

Aplicar el enfoque a análisis forense digital:

- Identificación de familias en incidentes de seguridad
- Clustering de muestras desconocidas
- Trazabilidad de variantes de amenazas

5.6.5. Investigación en Interpretabilidad

Explicabilidad Mejorada

Desarrollar métodos más sofisticados para interpretar decisiones:

- Análisis de sensibilidad local mediante perturbaciones
- Identificación de características mínimas necesarias para clasificación
- Generación de explicaciones en lenguaje natural para analistas

Extracción de Conocimiento Experto

Utilizar los modelos entrenados para extraer conocimiento sobre diferencias estructurales entre familias que pueda informar análisis manual de malware.

5.6.6. Benchmarking y Estandarización

Protocolo de Evaluación Estándar

Contribuir al establecimiento de protocolos de evaluación estandarizados para la comunidad:

- Definición de splits de entrenamiento/validación/prueba reproducibles
- Métricas estandarizadas considerando costos asimétricos (falsos negativos vs. falsos positivos)
- Datasets de referencia actualizados periódicamente

Challenge y Competición

Organizar competiciones académicas para impulsar avances en el campo, similar a challenges en visión por computador o NLP.

5.7. Implicaciones para la Ciberseguridad

Los resultados de este proyecto tienen implicaciones significativas para la práctica de la ciberseguridad:

5.7.1. Para Profesionales de Seguridad

- **Automatización:** Reducción del esfuerzo manual en análisis de grandes volúmenes de muestras sospechosas.
- **Velocidad:** Clasificación en tiempo prácticamente real permite respuesta más rápida a incidentes.
- **Escalabilidad:** Capacidad de procesar cantidades masivas de datos sin incremento lineal en recursos humanos.

5.7.2. Para Desarrolladores de Soluciones de Seguridad

- **Complementariedad:** Los métodos basados en Deep Learning pueden complementar (no reemplazar) soluciones tradicionales.
- **Adaptabilidad:** Los modelos pueden reentrenarse periódicamente para adaptarse a nuevas amenazas.
- **Multi-modal:** Posibilidad de fusionar análisis visual con otras técnicas para detección más robusta.

5.7.3. Para la Investigación Académica

- **Fundamentos sólidos:** Este trabajo proporciona evidencia experimental sobre la viabilidad del enfoque visual.
- **Dirección prometedora:** Se identifican múltiples líneas de investigación futuras con potencial impacto.

- **Metodología reproducible:** El pipeline implementado facilita la extensión y comparación con nuevos métodos.

5.8. Lecciones Aprendidas

Durante el desarrollo de este proyecto se adquirieron varias lecciones valiosas:

5.8.1. Técnicas

- **Importancia del preprocesamiento:** La calidad de las imágenes generadas impacta significativamente el rendimiento final.
- **Balance entre complejidad y datos:** Modelos muy profundos pueden no ser necesarios con datasets limitados.
- **Regularización esencial:** Dropout y data augmentation son críticos para evitar overfitting en este dominio.
- **Transfer learning valioso:** Las características de bajo nivel de ImageNet son sorprendentemente transferibles.

5.8.2. Prácticas

- **Experimentación sistemática:** La variación controlada de hiperparámetros es esencial para optimización.
- **Validación rigurosa:** La evaluación en conjunto de prueba separado es imprescindible para estimaciones realistas.
- **Interpretabilidad importante:** La capacidad de explicar decisiones es crucial para adopción en seguridad.

5.9. Consideraciones Éticas

El desarrollo de herramientas automatizadas de análisis de malware plantea consideraciones éticas importantes:

5.9.1. Uso Responsable

Los modelos y técnicas desarrollados deben emplearse exclusivamente para:

- Defensa legítima de sistemas y redes
- Investigación académica con fines educativos
- Análisis forense en el contexto de incidentes de seguridad

Nunca deben utilizarse para:

- Desarrollo de nuevas amenazas
- Evasión de sistemas de seguridad con intención maliciosa
- Ataques a sistemas sin autorización explícita

5.9.2. Transparencia

Es importante mantener transparencia sobre:

- Limitaciones de los modelos (tasas de falsos negativos/positivos)
- Datasets utilizados y sus sesgos inherentes
- Condiciones bajo las cuales los resultados son válidos

5.9.3. Privacidad y Confidencialidad

Al trabajar con muestras de malware reales, se debe:

- Proteger cualquier información sensible que puedan contener los ejecutables
- Cumplir con regulaciones de manejo de código malicioso
- Evitar diseminación no controlada de muestras activas

5.10. Reflexiones Finales

La clasificación automatizada de malware mediante Deep Learning representa un área de investigación madura y prometedora en la intersección de inteligencia artificial y ciberseguridad. Este proyecto ha demostrado que el enfoque basado en análisis visual de ejecutables es técnicamente viable y puede alcanzar niveles de rendimiento competitivos con el estado del arte.

Sin embargo, es fundamental reconocer que ninguna solución única resolverá por completo el problema de la detección de malware. Las amenazas continúan evolucionando, y los atacantes adaptan constantemente sus técnicas. Por lo tanto, los sistemas efectivos de ciberseguridad requieren enfoques en capas (defense in depth) que combinen múltiples técnicas complementarias.

El Deep Learning, y específicamente las CNN aplicadas a representaciones visuales, constituyen una herramienta poderosa en el arsenal del defensor, pero deben emplearse en conjunto con:

- Análisis heurístico y basado en firmas
- Detección comportamental
- Inteligencia de amenazas
- Supervisión humana experta

La investigación futura debe enfocarse no solo en mejorar la precisión de los modelos, sino también en su robustez, interpretabilidad, y aplicabilidad práctica en entornos de producción con requisitos estrictos de latencia y confiabilidad.

5.11. Conclusión

Este proyecto ha explorado exitosamente la aplicación de Redes Neuronales Convolucionales para la clasificación de familias de malware mediante análisis visual de

ejecutables. Los resultados experimentales confirman la hipótesis inicial: las CNN son capaces de aprender automáticamente representaciones discriminativas que permiten clasificación efectiva de malware.

Se han realizado contribuciones tanto técnicas (evaluación comparativa exhaustiva, análisis de generalización, estudio de interpretabilidad) como prácticas (identificación de limitaciones reales, recomendaciones para despliegue).

Las direcciones de trabajo futuro identificadas ofrecen oportunidades prometedoras para avanzar el estado del arte en este campo crítico para la seguridad de sistemas informáticos modernos.

En última instancia, este trabajo contribuye a la creciente evidencia de que las técnicas de Deep Learning representan una herramienta valiosa y cada vez más madura para abordar desafíos complejos en ciberseguridad, con potencial para impacto real en la protección de infraestructuras digitales frente a amenazas en constante evolución.

This page intentionally left blank.