El conjunto de datos de malware de Microsoft, empleado en la competición Kaggle del 2015 para clasificación de malware, se destaca por su considerable tamaño de casi medio terabyte. Este conjunto consta de 21,741 muestras de malware divididas en conjuntos de entrenamiento y prueba, con 10,868 y 10,873 muestras, respectivamente. Cada muestra pertenece a una de las 9 familias de malware identificadas.

1. Ramnit
2. Lollipop
3. Kelihos\_ver3
4. Vundo
5. Simda
6. Tracur
7. Kelihos\_ver1
8. Obfuscator.ACY
9. Gatak

Las "familias de malware" no están distribuidas equitativamente. Son grupos de programas maliciosos que comparten características y comportamientos similares. Los programas maliciosos dentro de una familia a menudo comparten un código base común o una funcionalidad específica.

Cada muestra cuenta con archivos .bytes y .asm asociados, que representan la información binaria y el código desensamblado, respectivamente. En la competición Kaggle, los participantes tuvieron acceso a estos archivos para desarrollar modelos de aprendizaje automático que clasificaran las muestras en sus respectivas familias de malware. La diversidad de técnicas de obfuscación y evasión utilizadas por los desarrolladores de malware desafía la efectividad de los modelos.

El artículo resalta que, a pesar de los desafíos, el modelo propuesto basado en una red neuronal convolucional (CNN) logró un rendimiento excepcional. La arquitectura CNN demostró su capacidad al superar incluso a los modelos ganadores de la competición en términos de precisión. Este éxito subraya el potencial del aprendizaje profundo, específicamente de las CNN, en la ciberseguridad, proporcionando resultados prometedores en la clasificación de muestras de malware.

Archivo .byte

00401000 56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08

00401010 BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC

00401020 C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC

00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44

00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6

00401050 5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC CC

La primera columna son las direcciones de memoria y de la 2-17 muestran los valores en formato hexadecimal que están almacenados en esas direcciones de memoria.

Cada **archivo .byte** en el conjunto de datos de entrenamiento contiene una tabla donde la primera columna representa direcciones de memoria. Las columnas de la 2 a la 17 contienen los "bytes en esa dirección", que se refiere a los valores específicos almacenados en direcciones de memoria consecutivas. Son valores específicos en formato hexadecimal, que se pueden convertir a su equivalente decimal para obtener la representación numérica de cada byte. Por ejemplo:

56: Representa el valor hexadecimal 56, que es equivalente a 86 en decimal.

8D: Representa el valor hexadecimal 8D, que es equivalente a 141 en decimal.

44: Representa el valor hexadecimal 44, que es equivalente a 68 en decimal.

En el contexto de la detección de malware, el análisis de estos bytes implica buscar patrones, secuencias o firmas que puedan indicar la presencia de código malicioso. Los modelos de aprendizaje automático, como las redes neuronales convolucionales (CNN), las redes neuronales recurrentes (RNN) o los autoencoders, pueden entrenarse para identificar estos patrones y ayudar en la clasificación de archivos como benignos o maliciosos.

Eso es lo que incluye cada archivo .byte y luego cada uno de ellos, tiene asociado el tipo de malware al que hace referencia en el documento **trainLabels**.

En el archivo sampleSubmission, sale el formato en el que se tiene que entregar las predicciones propuestas. El nombre del archivo debe mantenerse, lo que cambia son las predicciones y hay que comprobar que el sumatorio de 1.

El articulo que usa este dataset se resume en:

Te visualiza y explica como pasar de los datos a imagenes. Además te dice como sería el modelo de CNN con las capas y dimensiones. Por ultimo te dice la informacion necesaria como el indice de aprendizaje y las epochs necesarias para obtener el 98,7 \% de acurrancy

<https://simba.fdi.ucm.es/jupyter/user/pablji04/lab>

screen, ejecuto el programa(python3 ….py) y luego crt a z para salir de la pantalla y volver a la pantalla anterior. para volver a la pantalla que se esta ejecutando pongo screen -r : man screen

Elección del tamaño 224x224 de las imágenes reconocimiento tipo de malware

La elección del tamaño de las imágenes de 224x224 píxeles en la arquitectura de esta ConvNet se basa en varios factores, incluidos los siguientes:

Compatibilidad con la Red Preentrenada (Transfer Learning): La arquitectura de la ConvNet puede haber sido diseñada originalmente para trabajar con imágenes de 224x224 píxeles como parte de un enfoque de transfer learning. En la práctica, es común utilizar arquitecturas preentrenadas en conjuntos de datos grandes (como ImageNet) y adaptarlas para tareas específicas.

Requisitos de Memoria y Computación: Imágenes más grandes pueden requerir más memoria y recursos computacionales. Limitar el tamaño de las imágenes a 224x224 ayuda a gestionar eficientemente los recursos disponibles durante el entrenamiento y la inferencia.

Tamaño Suficiente para Características Relevantes: Aunque 224x224 puede parecer relativamente pequeño, es lo suficientemente grande para capturar características importantes de las imágenes. Además, el uso de capas convolucionales y técnicas como el max-pooling permite a la red aprender representaciones jerárquicas de características a diferentes escalas.

Consistencia con Conjuntos de Datos y Evaluaciones: Los conjuntos de datos y las métricas de evaluación, como en el caso del conjunto de datos ILSVRC, pueden estar diseñados específicamente para trabajar con imágenes de un tamaño particular. Utilizar imágenes de 224x224 garantiza la consistencia en la evaluación de modelos y comparaciones entre diferentes enfoques.

En resumen, la elección del tamaño de las imágenes es una combinación de consideraciones prácticas, compatibilidad con arquitecturas existentes y requisitos específicos del conjunto de datos y las tareas asociadas

pyenv local 3.10