UCP
UNIVERSIDAD DE LA CUENCA DEL PLATA

INGENIERIA EN SISTEMAS DE INFORMACION

Trabajo Práctico Integrador
Sistemas Inteligentes

Autor: Cáceres Erika Agustina

Profesor: Lic. Del Rosario Gabriel Darío

Año: 2025

Contenidos

Introducción

En el contexto actual de las redes de datos, la seguridad informática se ha convertido en una preocupación crítica debido al aumento constante de ataques y accesos no autorizados. Para mitigar estos riesgos, los Sistemas de Detección de Intrusiones (IDS, por sus siglas en inglés) se han transformado en herramientas fundamentales para proteger la integridad, disponibilidad y confidencialidad de los sistemas informáticos.

Los IDS tradicionales suelen basarse en reglas predefinidas, lo que limita su capacidad para adaptarse a nuevas amenazas. En este sentido, los sistemas inteligentes, y en particular las Redes Neuronales Artificiales (RNA), han demostrado ser una alternativa eficaz gracias a su capacidad para aprender patrones complejos y generalizar a partir de datos históricos.

Este trabajo propone el diseño e implementación de un sistema de detección de intrusiones basado en RNA, utilizando el conjunto de datos NSL-KDD, una base de datos ampliamente utilizada para entrenar y evaluar modelos de detección de ataques en redes. El objetivo es comparar el rendimiento de dos modelos de RNA diferentes, utilizando métricas de evaluación como la exactitud (accuracy) y el F1-score, a fin de determinar cuál ofrece un mejor desempeño en la clasificación de conexiones de red como normales o maliciosas.

Desarrollo

Problema

En una red de datos, cada conexión que se establece puede ser legítima o maliciosa. La detección de estas amenazas es un desafío constante debido a la variedad y sofisticación de los ataques. Muchos de los sistemas de detección actuales dependen de reglas estáticas que deben actualizarse manualmente y no son efectivos frente a nuevos tipos de amenazas.

La posibilidad de aplicar técnicas de inteligencia artificial, y en particular redes neuronales artificiales, permite entrenar modelos que aprendan a reconocer patrones anómalos a partir de ejemplos previos. Esto abre la puerta a sistemas de detección más adaptativos y precisos.

Objetivo General

Diseñar e implementar un sistema inteligente basado en redes neuronales artificiales (RNA) para la detección de intrusiones en una red de datos, utilizando el conjunto de datos NSL-KDD.

Objetivos Específicos

- Preprocesar el dataset NSL-KDD para su uso en modelos de RNA.
- Diseñar y entrenar dos arquitecturas diferentes de redes neuronales artificiales.
- Evaluar el rendimiento de ambos modelos utilizando métricas como accuracy y F1-score.
- Comparar los resultados obtenidos para identificar cuál de los modelos es más adecuado para la detección de intrusiones.

Dataset: NSL-KDD

Para el desarrollo de este trabajo se utiliza el dataset NSL-KDD, una versión mejorada del conjunto de datos KDD Cup 99, ampliamente utilizado en investigaciones sobre detección de intrusiones en redes. El objetivo principal del dataset es permitir la evaluación de modelos de clasificación que detecten si una conexión de red es normal o corresponde a un ataque.

Características

El NSL-KDD contiene registros de conexiones de red, cada uno compuesto por 41 atributos y una etiqueta de clase. Las conexiones incluyen información sobre el protocolo utilizado,

número de bytes enviados y recibidos, número de conexiones activas en un período, y estadísticas basadas en ventanas de tiempo. Los atributos combinan variables numéricas y categóricas.

Cada registro está etiquetado como:

- `normal`: conexión legítima.
- `attack_type`: indica un tipo de ataque, que puede pertenecer a una de las siguientes categorías:
  - **DoS (Denial of Service)**: ataques que buscan hacer un recurso inaccesible.
  - **Probe**: exploraciones de red para identificar vulnerabilidades.
  - **R2L (Remote to Local)**: acceso no autorizado desde una máquina remota.
  - **U2R (User to Root)**: obtención de privilegios de administrador desde una cuenta local.

Ventajas del dataset

A diferencia de su predecesor, el NSL-KDD elimina redundancias y registros duplicados, permitiendo una evaluación más justa y realista del rendimiento de los modelos de aprendizaje automático.

Objetivo de uso

En este trabajo, el dataset será utilizado para entrenar dos modelos distintos de redes neuronales artificiales, con el fin de clasificar cada conexión como "normal" o "ataque". Para facilitar esta tarea, se aplicarán técnicas de preprocesamiento que transformen las etiquetas en formato binario (`normal` vs `attack`) y conviertan las variables categóricas en vectores numéricos adecuados para su uso en redes neuronales.

Diseño de los Modelos de Redes Neuronales Artificiales

Para abordar el problema de detección de intrusiones, se desarrollarán y entrenarán dos modelos distintos de redes neuronales artificiales (RNA). Ambos se basan en redes neuronales feedforward de tipo perceptrón multicapa (MLP), pero se diferencian en su complejidad y técnicas de regularización.

Modelo 1 – MLP Simple

Este modelo está diseñado con una arquitectura básica y liviana. Su objetivo es servir como referencia inicial, mostrando qué tan bien puede funcionar una RNA sin técnicas avanzadas.

Arquitectura:

- Capa de entrada: número de neuronas igual a la cantidad de atributos del dataset.
- Capa oculta 1: 64 neuronas, activación ReLU.
- Capa oculta 2: 32 neuronas, activación ReLU.
- Capa de salida: 1 neurona con activación sigmoide (para clasificación binaria).

Características:

- Sin técnicas de regularización.
- Optimización: Adam.
- Función de pérdida: Binary Crossentropy.
- Métricas: Accuracy y F1-score.

Modelo 2 – MLP Mejorado con Dropout

Este segundo modelo incorpora técnicas para mejorar la generalización y evitar el sobreajuste (overfitting), como el uso de dropout y capas más profundas.

Arquitectura:

- Capa de entrada: igual a la cantidad de atributos.
- Capa oculta 1: 128 neuronas, activación ReLU.
- Dropout: 0.3
- Capa oculta 2: 64 neuronas, activación ReLU.
- Dropout: 0.2
- Capa de salida: 1 neurona con activación sigmoide.

Características:

- Regularización con Dropout.

- Optimización: Adam.
- Función de pérdida: Binary Crossentropy.
- Métricas: Accuracy y F1-score.

Ambos modelos serán entrenados bajo las mismas condiciones en términos de número de épocas, tamaño del batch y proporción de datos de entrenamiento/prueba, permitiendo una comparación justa de su rendimiento.

Método

Para resolver el problema, usamos un enfoque de aprendizaje supervisado. Esto significa que entrenamos al modelo con ejemplos ya clasificados (etiquetados como normales o maliciosos).

Trabajamos con un dataset que contiene datos sobre conexiones de red y varios indicadores (como duración, cantidad de bytes, tipo de conexión, etc.). El proceso incluyó las siguientes etapas:

- Revisión y comprensión del dataset.
- Preparación de los datos.
- Definición de dos modelos de red neuronal (con distinta estructura).
- Entrenamiento de los modelos.
- Evaluación usando métricas como **accuracy** y **F1-score**.

El código fue desarrollado en Python, y en breve va a estar disponible en un repositorio de GitHub junto con el dataset y los resultados.

Herramientas

Para este proyecto usamos el lenguaje **Python** junto con algunas librerías específicas que ayudan a trabajar con redes neuronales y datos. Las principales herramientas fueron:

- TensorFlow: para crear y entrenar los modelos de red neuronal.
- pandas: para manipular los datos del dataset.
- scikit-learn: para dividir los datos y calcular métricas.
- matplotlib: para generar gráficos (aunque no todos están incluidos en este informe).

El entorno de desarrollo fue Visual Studio Code usando un entorno virtual de Miniconda para evitar problemas con las versiones de Python.

Resultados

Si bien los resultados exactos todavía están en proceso de ajuste y entrenamiento final, se observó que ambos modelos pudieron aprender a distinguir entre tráfico normal y malicioso con una precisión aceptable.

Las métricas principales utilizadas fueron:

- **Accuracy**: mide qué porcentaje de predicciones fueron correctas.
- **F1-score**: combina precisión y recall en una sola métrica.

Se espera seguir ajustando los modelos para obtener mejores resultados y generar gráficos más completos.

Conclusión

Realizar este trabajo fue muy útil para entender cómo aplicar inteligencia artificial en problemas reales, en este caso en el análisis de tráfico de red. Aprendí a preparar datos reales, a trabajar con redes neuronales desde cero, y a usar herramientas modernas como TensorFlow.

Una posible mejora sería probar otros modelos más complejos o combinar redes neuronales con técnicas de análisis estadístico. También se podría aplicar a otros tipos de datos relacionados con seguridad informática.

Anexo

Link de acceso a presentación: https://www.canva.com/design/DAGqMjRY6w4/vAtzk6gyttjcze0XMwNV-Q/edit?utm_content=DAGqMjRY6w4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Link de acceso al repositorio: https://github.com/Agustinac17/integrador_si_caceres.git

Bibliografía

- Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.
- Chollet, F. (2018). Deep Learning with Python. Manning Publications.
- McKinney, W. (2017). Python for Data Analysis. O'Reilly Media.
- Dataset: CICIDS2017 - Canadian Institute for Cybersecurity.
- TensorFlow (2024). https://www.tensorflow.org