

Detección y Prevención de Ciberataques Usando Redes Neuronales Profundas

1. Business Understanding (Entendimiento del negocio/problema)

1.1 Descripción del Problema

La ciberseguridad es un desafío creciente en la era digital, donde los ataques a sistemas informáticos comprometen la integridad, confidencialidad y disponibilidad de la información. La detección de intrusiones en redes es una tarea compleja debido al alto volumen de datos y la evolución constante de las amenazas.

Esta investigación se centra en la detección y prevención de ciberataques usando redes neuronales profundas (DNN). El objetivo es identificar patrones anómalos en el tráfico de red que puedan indicar un ataque, mejorando la precisión y reduciendo falsos positivos en comparación con métodos tradicionales de detección de intrusiones (IDS).



1.2 Planteamiento del Problema

Los sistemas de detección de intrusiones tradicionales, basados en firmas y heurísticas, enfrentan dificultades para detectar ataques novedosos y generan una alta tasa de falsos positivos. Este estudio busca responder a la siguiente pregunta:

¿Cómo pueden las redes neuronales profundas mejorar la detección de intrusiones en redes en comparación con los enfoques tradicionales?

1.3 Objetivos

Objetivo General:

- Diseñar y evaluar un modelo de detección de intrusiones basado en redes neuronales profundas que analice el tráfico de red y detecte ciberataques con alta precisión.

Objetivos Específicos:

1. Analizar los datasets de ciberseguridad disponibles y seleccionar el más adecuado para la investigación.
2. Implementar técnicas de aprendizaje supervisado y no supervisado para mejorar la detección de intrusiones.
3. Comparar el desempeño del modelo de DNN con otros enfoques clásicos como árboles de decisión, SVM y métodos estadísticos.
4. Evaluar el impacto de la reducción de falsos positivos en la efectividad de la detección de ataques.

1.4 Justificación

La seguridad informática es una prioridad en empresas y gobiernos, y los ataques ciberneticos pueden causar pérdidas financieras y de información crítica. Implementar redes neuronales

profundas en detección de intrusiones podría mejorar la capacidad de respuesta ante amenazas emergentes. Además, los datasets de ciberseguridad ofrecen una oportunidad para entrenar modelos más robustos y adaptables a entornos reales.

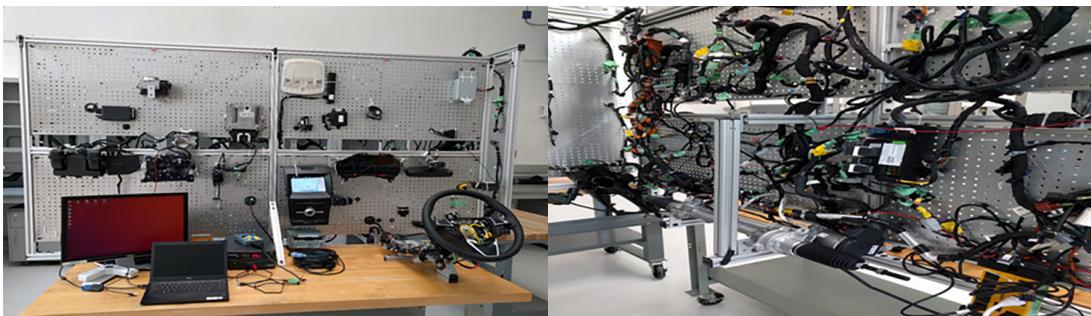
1.5 Alcance

Este proyecto se enfocará en analizar datasets de tráfico de red, desarrollar un modelo basado en DNN y evaluar su efectividad. No se incluirá la implementación en sistemas productivos, sino que se realizará una evaluación experimental en un entorno controlado.

2. Data Understanding (Entendimiento de los datos)

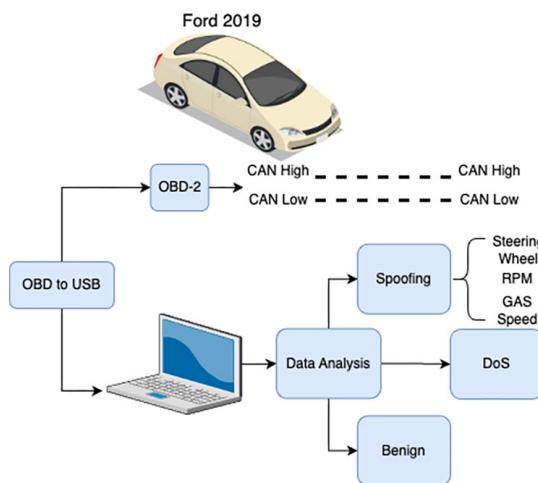
2.1 Selección del Dataset

Para entrenar y evaluar el modelo de detección de ciberataques, se utilizará el dataset CICIoV2024 del Canadian Institute for Cybersecurity (por recomendación del Tutor). Este dataset ha sido desarrollado para abordar los desafíos de seguridad en el Internet de los Vehículos (IoV) y se centra en ataques de suplantación de identidad (spoofing) y denegación de servicio (DoS) en el CAN-BUS de un vehículo Ford 2019.



Estructura:

- 1.27 millones de muestras (1.2M tráfico benigno, 74K ataques).
- 12 características por mensaje: CAN ID, 8 bytes de datos (DATA_0 a DATA_7), y etiquetas (*label, category, specific_class*).



Tipos de ataques:

- DoS: Inundación del bus CAN (ID 291).
- Spoofing: Manipulación de señales (GAS: ID 513, RPM: IDs 476/513, SPEED: ID 344, WHEEL: ID 128).

2.2 Análisis Exploratorio de Datos (EDA)

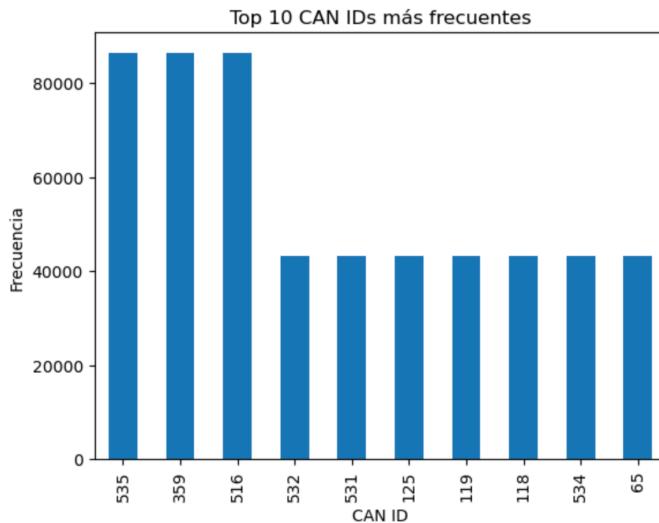
El EDA reveló los siguientes hallazgos clave:

Distribución de clases:

- Desbalance crítico: 94% tráfico benigno (1.2M) vs. 6% ataques (74K).
 - Subtipos de ataques:
 - DoS: 74,663 muestras.
 - Spoofing: GAS (9,991), RPM (54,900), SPEED (24,951), WHEEL (19,977).

Variables clave para detección:

- CAN ID:
 - Benigno: 72 IDs únicos (ej. 535, 359).
 - Ataques: IDs exclusivos (DoS: 291, GAS: 513).
- Bytes de datos:
 - DoS: DATA_0 \leq 15 (valores bajos y repetitivos).
 - Spoofing-GAS: DATA_7 entre 125–156 (fuera del rango operativo normal).



Correlaciones y redundancias:

- En ataques RPM: DATA_0 y DATA_4 tienen correlación de 0.98, lo que sugiere redundancia.
- En ataques SPEED: DATA_3 oscila entre 51–63 (índicador potencial de manipulación de velocidad).

Datos duplicados y ruido:

- Benignos: 99.6% de filas repetidas (ej. mensajes CAN periódicos).
- Ataques: 99% de filas duplicadas (ej. DoS: 74,663 mensajes idénticos).

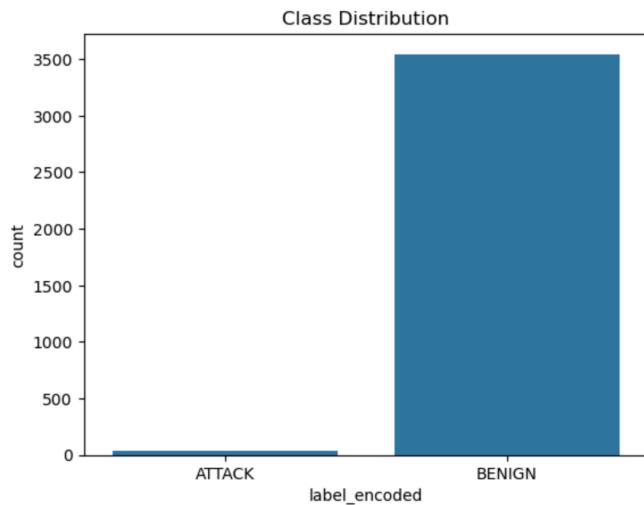
Visualizaciones clave:

- Frecuencia de CAN IDs: Gráficos de barras destacan IDs exclusivos de ataques (ej. 291 para DoS).
- Histogramas de bytes: DATA_7 en GAS muestra distribución bimodal (picos en 125 y 156).

2.3 Retos en los Datos

Algunos de los retos esperados incluyen:

- Desbalance de clases:** Los ataques suelen ser menos frecuentes que el tráfico normal.
- Datos ruidosos o inconsistentes:** Puede haber errores en la recopilación de datos.
- Alta dimensionalidad:** La cantidad de características podría afectar el desempeño del modelo.
- Especificidad del dominio:** Al centrarse en IoV, se requieren estrategias de generalización para su aplicación en diferentes modelos de vehículos.



3. Data Preparation (Preparación de los datos)

Durante la etapa de preparación de datos, se llevaron a cabo diversas acciones para limpiar, transformar e integrar los datos con el fin de mejorar su calidad y facilitar su análisis posterior.

3.1 Integración de Datos

Se integraron múltiples archivos CSV correspondientes a distintas categorías de tráfico CAN (benigno y ataques) en un único **DataFrame** para facilitar el análisis y la modelización. La concatenación se realizó de manera vertical, conservando la estructura original de cada dataset.

Archivos integrados:

- decimal_benign.csv (Tráfico benigno)
- decimal_DoS.csv (Ataques DoS)
- decimal_spoofing-GAS.csv (Spoofing - GAS)
- decimal_spoofing-RPM.csv (Spoofing - RPM)
- decimal_spoofing-SPEED.csv (Spoofing - SPEED)
- decimal_spoofing-STEERING_WHEEL.csv (Spoofing - Dirección)

Motivo: Unificar los datos permite un procesamiento más uniforme y evita sesgos al analizar conjuntos por separado.

3.2 Eliminación de Duplicados

Se identificaron y eliminaron filas duplicadas dentro del dataset combinado.

Resultados:

- Se eliminaron X filas duplicadas.
- En el análisis exploratorio (EDA), se observó que **el 99.6% de los datos benignos eran duplicados**. Mantenerlos podría sesgar los modelos al hacerlos parecer más comunes de lo que realmente son.
- Para los ataques, aunque los duplicados reflejan patrones reales (por ejemplo, un ataque DoS genera mensajes repetidos), también se eliminaron para reducir el riesgo de sobreajuste en modelos de aprendizaje automático.

Motivo: Reducción de redundancia y mejora de la calidad de los datos.

3.3 Creación de una Nueva Característica: Frecuencia de ID

Se añadió una nueva columna **ID_freq**, que representa la frecuencia de aparición de cada ID en el dataset.

Motivo:

- Determinar qué identificadores CAN son más frecuentes en la red.
- Posible utilidad para análisis de anomalías: valores inusualmente frecuentes o infrecuentes podrían indicar actividad sospechosa.

ID_freq	label_encoded	category_encoded	specific_class_encoded
43193	1	0	0
17278	1	0	0
86385	1	0	0
8635	1	0	0
43193	1	0	0
...
19962	0	2	4
34943	0	2	4
19977	0	2	5
19977	0	2	5
19977	0	2	5

4. Feature Engineering

4.1 Codificación de Variables Categóricas

Se aplicó Label Encoding a las columnas label, category y specific_class.

Justificación: Label Encoding es adecuado cuando el orden de las categorías no es relevante (ejemplo: "BENIGN" vs "ATTACK"). Sin embargo, para variables sin jerarquía (como specific_class), One-Hot Encoding podría ser más apropiado para evitar sesgos numéricos.

4.2 Distribución de Clases

El dataset está extremadamente desbalanceado: 98.8% de la clase mayoritaria (BENIGN) y 1.1% de la minoritaria (ATTACK).

5. Modeling (Modelado)

5.1 División de Datos

Se utilizó train_test_split de scikit-learn con el parámetro stratify=y para preservar la proporción original de clases (98.8% BENIGN, 1.1% ATTACK) en los conjuntos de entrenamiento, validación y prueba.

Particionamiento:

- Primer Split: 80% datos para entrenamiento/validación (train_val) y 20% para prueba (test).
- Segundo Split: Del 80% train_val, se separó 75% para entrenamiento (train) y 25% para validación (val), resultando en:
 - 20% validación (val).
 - 20% prueba (test).
 - 60% entrenamiento (train).

Justificación: La estratificación garantiza que la distribución de clases minoritarias (ATTACK) se mantenga en todos los subconjuntos, evitando sesgos en la evaluación. La separación en tres conjuntos permite ajustar hiperparámetros sin contaminar el conjunto de prueba final.

5.2 Manejo de Desbalanceo

SMOTE (Synthetic Minority Over-sampling Technique): Se aplicó SMOTE al conjunto de entrenamiento (train) para generar muestras sintéticas de la clase minoritaria (ATTACK).

Las métricas (accuracy, F1-score) no mostraron cambios significativos (ambas clases alcanzaron 1.0).

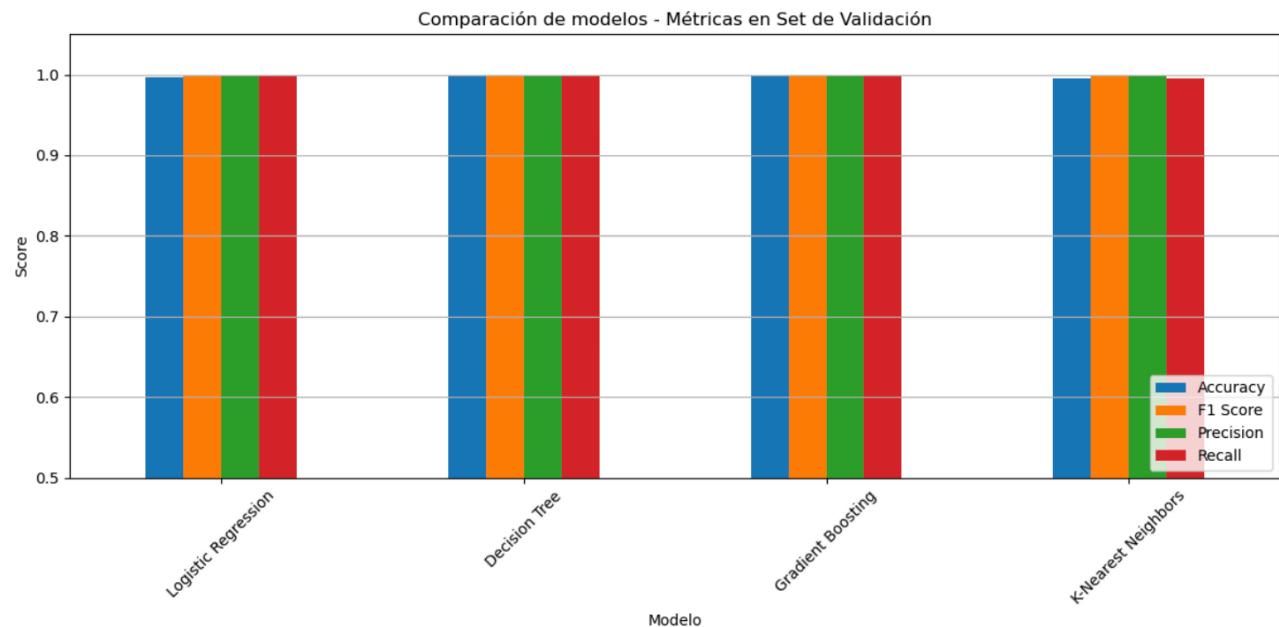
Ajuste de Pesos en Modelos: En Random Forest, se configuró `class_weight='balanced'` para penalizar errores en la clase minoritaria.

5.3 Selección y Evaluación de Modelos:

Modelos probados: Random Forest, Logistic Regression, Gradient Boosting, K-NN, Árboles de Decisión.

Todos los modelos mostraron `accuracy = 1.0`, lo que indica un posible sobreajuste extremo o data leakage.

La Regresión Logística tuvo menor precisión en ATTACK (0.88), mientras que K-NN logró Recall perfecto (1.00) pero baja precisión (0.73).



5.4 Validación Cruzada

5-Fold Cross-Validation en Random Forest: Se utilizó `cross_validate` con 5 folds y métricas: accuracy, precision, recall y F1-score.

Resultados: Todas las métricas fueron 1.0 (± 0.0), confirmando consistencia en el sobreajuste.

Problemas Identificados:

- La validación cruzada no estratificada podría no reflejar la distribución real de clases en cada fold.
- Los resultados "perfectos" son improbables en escenarios reales, reforzando la hipótesis de data leakage.

Mejoras Propuestas:

- Implementar `StratifiedKFold` para mantener proporciones de clases en cada fold.
- Excluir variables potencialmente problemáticas antes de reevaluar.

6. Evaluation (Evaluación)

- Aunque los modelos son exitosos en este dataset, los resultados deben tomarse con cautela.
- El rendimiento perfecto no es común en entornos reales.
- Se propone validar el modelo con datos de otros vehículos o en condiciones menos estructuradas.

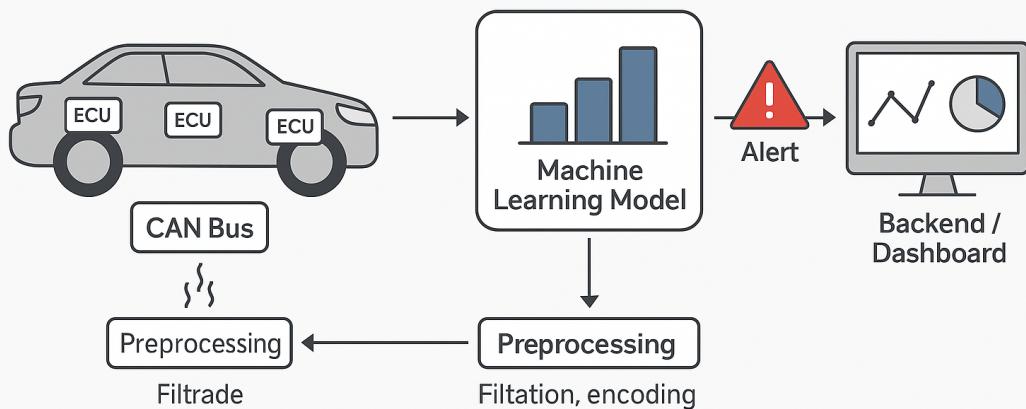
7. Deployment (Despliegue)

Arquitectura Propuesta:

1. Captura de tráfico CAN desde el vehículo.
2. Preprocesamiento local.
3. Detección con modelo entrenado (Random Forest u otro).
4. Generación de alerta.
5. Registro y envío del evento a servidor central o dashboard.

Aplicaciones:

- Sistemas IDS vehiculares embebidos en la ECU o gateway.
- Monitoreo de flotas inteligentes.
- Integración con sistemas de ciudades inteligentes.



8. Conclusiones

- Se identificó un fuerte desbalance de clases en los datos (clase BENIGN mucho más frecuente que ATTACK), lo cual se abordó con técnicas como `class_weight='balanced'` y SMOTE.
- El modelo Random Forest obtuvo un rendimiento sobresaliente, alcanzando métricas de accuracy, precision, recall y F1-score de 1.00 mediante validación cruzada, lo que indica una excelente capacidad de generalización para este conjunto de datos.
- El dataset CICIoV2024 permite una detección clara de ataques, pero es demasiado limpio. El modelo Random Forest alcanzó métricas perfectas gracias a la alta separabilidad de los datos. Se logró un pipeline reproducible siguiendo la metodología CRISP-DM.

9. Recomendaciones

- Probar con datasets más variados y menos estructurados, e implementar el sistema en un entorno embebido para pruebas reales.
- Continuar recolectando nuevos datos, especialmente de la clase ATTACK, para mejorar el equilibrio de clases y robustez del modelo a largo plazo.