Curso: Deep Learning
Semestre: 2022-01
Estudiante: Omar Augusto Roa Romero
Correo electrónico: omar.roa@udea.edu.co
Programa: Maestría en Ingeniería de Telecomunicaciones
Entrega: 2

NOTEBOOKS

Ubicación solicitada: https://github.com/omar-roa/deep-learning
Ubicación respaldo: https://drive.google.com/drive/folders/167AoM113FecfCMMi6x7-Qgzk66vEk9Tq?usp=sharing

01 - exploración de datos.ipynb

- Presentación de dataset
- Primera exploración
  * Revisión de características presentes
  * Conteo de observaciones por clase
  * Comparativo entre clases
  * Verificación de tipo de datos almacenados
- Análisis de dataset
  * Valores máximos y mínimos
  * Comparativo entre puerto de destino (Protocolo) y ataques
  * Mapa de calor de correlaciones
  * Correlaciones más altas entre características

Nota. Este cuaderno no fue posible guardarlo desde Colab a Github porque presentaba un error de demasiado tiempo renderizando. Fue agregado por carga normal a Github

02 - preprocesado.ipynb

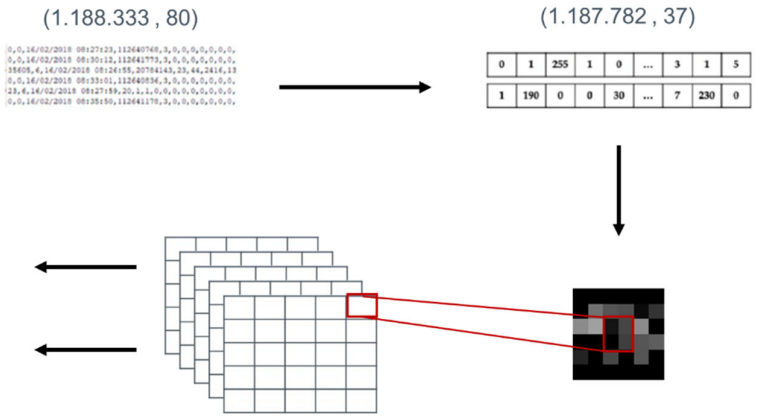
- Identificando y eliminando los datos nulos
- Identificando y eliminando los datos infinitos
- Características innecesarias
  * Identificación y eliminación de características constantes
  * Identificación de las características más representativas por el algoritmo Random Forest Importance.
  * Selección de las 36 características más importantes y eliminación de las restantes
- Desbalance entre BENIGNO y ATAQUES (2 clases)
  Se corre un modelo de clasificación clásico sin dataset balanceado y nuevamente con uno balanceado para revisar la matriz de confusión en ambos casos

03 - arquitectura de linea de base.ipynb

- Se importan los datasets ya procesados
  * Dataset depurado sin balancear (1.187.782 registros)
  * Dataset depurado y balanceado para 2 clases (780.004 registros)
- Se transforman las observaciones en imágenes 6x6 en escala de grises
- Se dividen los datos entre entrenamiento y prueba tanto para el escenario de 2 clases como para 5 clases.
- Modelos.
  * 2 clases con 1 capa convolución
  * 5 clases con 1 capa convolución
  * 5 clases con 1 capa convolución + dropout + 1 capa densa
  * 5 clases con 2 capa convolución + dropout/maxpooling + 1 capa densa

SOLUCIÓN

La solución aplicada se muestra gráficamente a continuación:



Un dataset original de 1.188.333 es procesado para eliminar datos atípicos: nulos (71) e infinitos (480), además se identifican algunas características constantes (10) y otras de menor importancia para un modelo de aprendizaje de máquina (33).

Luego se procede a convertir cada registro en una matriz 6x6 y escalar a valores entre 0 y 255 para representarlos como una imagen cuadrada en escala de grises.

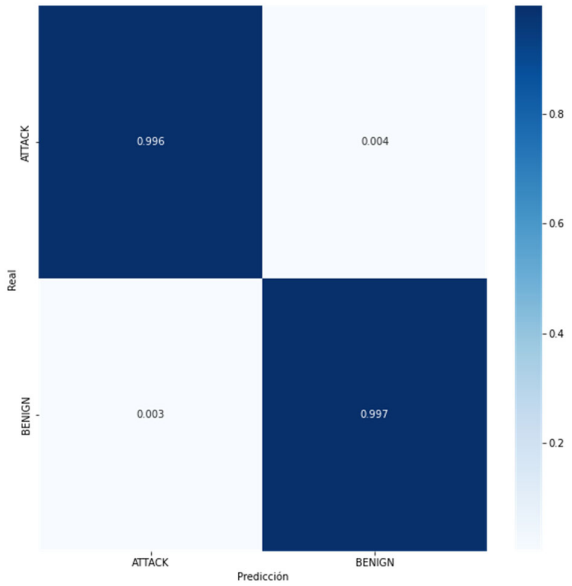
Ahora con un dataset de 1.187.782 imágenes cuadradas en escala de grises se ingresan a los modelos CNN. Es de interés revisar el desempeño del modelo para el caso binario (tráfico BENIGNO vs ATAQUE) y el caso multiclase (BENIGNO vs DDoS vs Brute Force vs XSS vs SQL Injection)

De manera intuitiva se plantean 4 modelos:

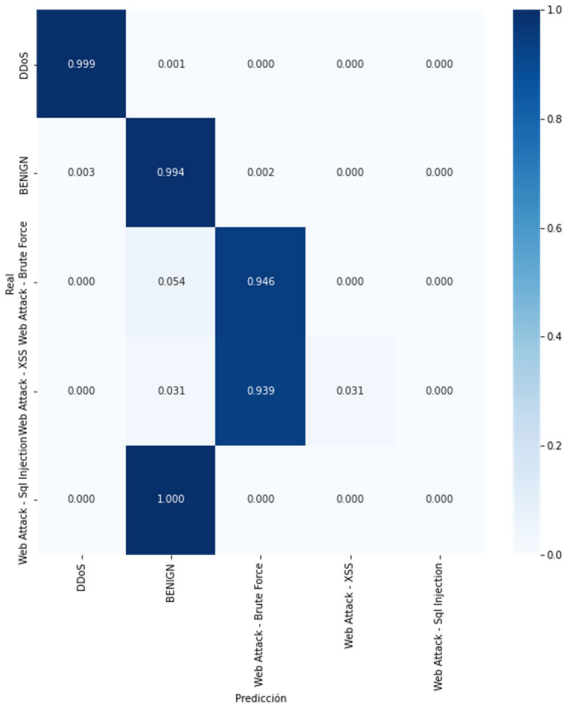
- Modelo A. 1 capa de convolución (15 de 3x3) para clasificación binaria. Este modelo lo entrenamos para realizar un primer acercamiento y tener valores referentes. Además, se aplica con el dataset no balanceado binario (BENIGNO vs ATAQUE) para identificar si es necesario correrlo con el dataset balanceado
- Modelo B. 1 capa de convolución (15 de 3x3) para clasificación multiclase. Ahora hacemos el primer acercamiento multiclase con la expectativa de la clase más desbalanceada presente (SQL Injection).
- Modelo C. 1 capa de convolución (15 de 3x3) + 1 capa densa (16) + Dropout (0.2) para clasificación multiclase. Busca validar si mejora el desempeño del modelo frente a las clases desbalanceadas.
- Modelo D. 1 capa de convolución (15 de 2x2) + 1 capa de convolución (60 de 2x2) + 1 capa densa (16) + Dropout (0.2) + MaxPooling (2x2) para clasificación multiclase. Busca validar si mejora el desempeño del modelo frente a las clases desbalanceadas.

RESULTADOS

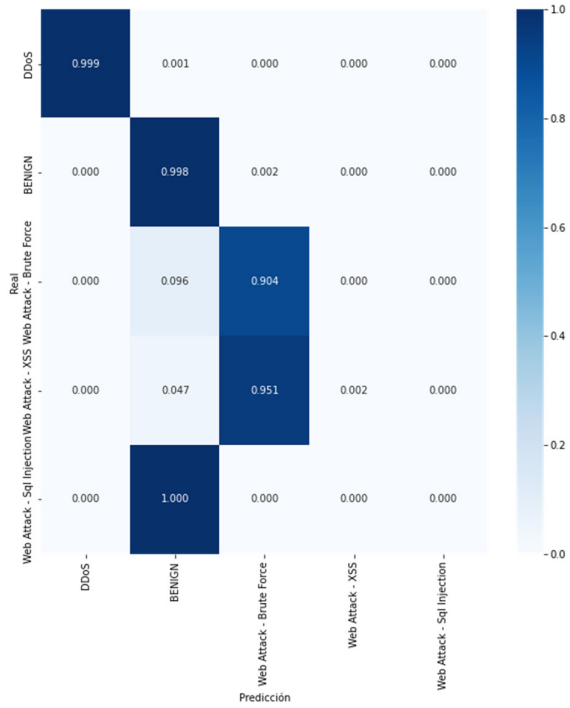
Una vez entrenados los 4 modelos se obtienen las siguientes matrices de confusión:



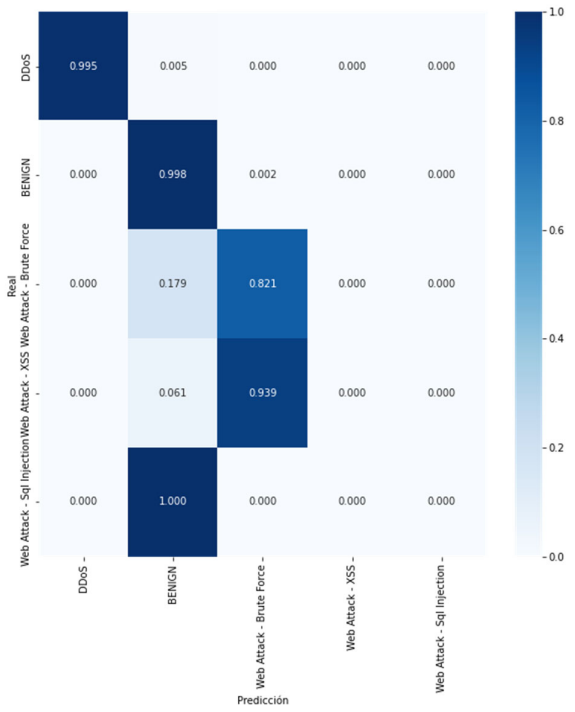
Matriz de confusión. Clasificación binaria (Modelo A)



Matriz de confusión. Clasificación multiclase (Modelo B)



Matriz de confusión. Clasificación multiclase (Modelo C)



Matriz de confusión. Clasificación multiclase (Modelo D)

Modelo	Clasificación	Parámetros	Accuracy	F1-Score
A	Binaria	1.232	0.996625	0.996175
B	Multiclase	2.855	0.994029	0.546193
C	Multiclase	8.891	0.996076	0.533337
D	Multiclase	4.796	0.994551	0.529526

Métricas de desempeño

En el escenario **binario**, aún con dataset en una proporción 2:1 entre las dos clases, presentó un desempeño destacado. Por ello, no vi conveniente realizar un modelo con el dataset balanceado para el caso binario.

En el escenario **multiclase**, era muy importante verificar tanto la matriz de confusión como la métrica F1-Score para entender el desempeño frente a las clases minoritarias. Como se Claramente F1-Score se va deteriorando a pesar de que las clases mayoritarias se mantienen casi estables en los True Positives. Claramente el impacto de tener la clase menor desbalanceada con respecto a la clase mayor en 13.000:1 no puede ser solucionado por el modelo. El dataset multiclase debe ser balanceado.

DATASET

El Dataset original se encuentra en https://www.kaggle.com/datasets/subhajournal/sdn-intrusion-detection

Una copia del Dataset original y los Datasets procesados están en el repositorio desde donde son cargados por los cuadernos

https://drive.google.com/drive/folders/167AoM113FecfCMMi6x7-Qgzk66vEk9Tq?usp=sharing

REFERENCIAS

B. Cao, et.al., “Network Intrusion Detection Model Based on CNN and GRU,” Applied Sciences, 12. 4184, 2022, doi:10.3390/app12094184.

J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, “CNN-Based Network Intrusion Detection against Denial-of-Service Attacks,” Electronics, vol. 9, no. 6, p. 916, Jun. 2020, doi:10.3390/electronics9060916.

A. Shaaban, E. Abd-Elwanis, and M. Hussein, “DDoS attack detection and classification via Convolutional Neural Network (CNN)”, 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), 2019, pp. 233-238, doi:10.1109/ICICIS46948.2019.9014826.

R. Vinayakumar, K. P. Soman and P. Poornachandran, “Applying convolutional neural network for network intrusion detection,” 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 1222-1228, doi:10.1109/ICACCI.2017.8126009.