UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

RED NEURONAL CONVOLUCIONAL CON ATENCIÓN JERÁRQUICA PARA DETECCIÓN DE INTRUSOS EVALUADO CON DATOS REALES

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

RODOLFO MARTÍNEZ CADENA

BAJO LA DIRECCIÓN DE:

DR. JOSÉ ADÁN HERNÁNDEZ NOLASCO

CUNDUACÁN, TABASCO, A: JUNIO 2025

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

RED NEURONAL CONVOLUCIONAL CON ATENCIÓN JERÁRQUICA PARA DETECCIÓN DE INTRUSOS EVALUADO CON DATOS REALES

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

RODOLFO MARTÍNEZ CADENA

BAJO LA DIRECCIÓN DE:

DR. JOSÉ ADÁN HERNÁNDEZ NOLASCO

CUNDUACÁN, TABASCO, A: JUNIO 2025

Declaración de Autoría y Originalidad

En la Ciudad de Cunduacán el día seis del mes de junio del año 2025, el que suscribe **Rodolfo Martínez Cadena**, alumno del Programa de la **Maestro en Ciencias de la Computación** con número de matrícula **232H21006**, adscrito a la **División Académica de Ciencias y Tecnologías de la Información**, de la Universidad Juárez Autónoma de Tabasco, como autor de la Tesis presentada para la obtención de Grado de Maestría y titulada **Red neuronal convolucional con atención jerárquica para detección de intrusos evaluado con datos reales**, dirigida por el Dr. José Adán Hernández Nolasco.

DECLARO QUE: La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Cunduacán, Tabasco a 6 de junio de 2025.

Estudiante: Rodolfo Martínez Cadena

UJAT
UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

2025 AÑO DE LA Mujer Indígena

Cunduacán, Tabasco, a 06 de junio de 2025
Oficio No. 0880/2025/DACYTI/D

Asunto: Autorización de impresión de Tesis

C. Rodolfo Martínez Cadena
Egresado de la Maestría en Ciencias de la Computación

En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento General de Estudios de Posgrado vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional ***"Red neuronal convolucional con atención jerárquica para detección de intrusos evaluado con datos reales"***, para presentar examen y obtener el Grado de Maestro en Ciencias de la Computación.

Sin otro particular, aprovecho la ocasión para enviarle un afectuoso saludo.

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

Atentamente

MTE. Oscar Alberto González González
Director

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

C.c.p. Dr. Eddy Arquímedes García Alcocer. - Encargado del Despacho de la Coordinación de Posgrado DACYTI
Archivo.
Consecutivo.

M.T.E. OAGG/EAGA

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690.
Cunduacán, Tabasco, México.
Tel: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870
E-mail: direccion.dacyti@ujat.mx

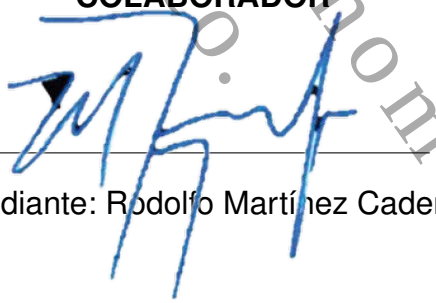
Carta de Cesión de Derechos

Villahermosa, Tabasco a 6 de junio de 2025.

Por medio de la presente manifiesto haber colaborado como AUTOR en la producción, creación y/o realización de la obra denominada: **Red neuronal convolucional con atención jerárquica para detección de intrusos evaluado con datos reales**.

Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención de que tenemos el derecho al reconocimiento como autores de la obra, y a la Universidad Juárez Autónoma de Tabasco mantendrá en un 100 % la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

COLABORADOR



Estudiante: Rodolfo Martínez Cadena

TESTIGOS

Índice general

Índice de figuras

Índice de tablas

Resumen

El rápido aumento de usuarios de internet y dispositivos conectados ha incrementado significativamente los riesgos de ciberseguridad, lo que hace necesaria la implementación de sistemas avanzados de detección de intrusos (IDS). En este trabajo, se presenta **H.A.L.C.CO.N** (Hierarchical Attention-based Loss Equalization with CatBoost-enhanced Convolutional Neural Network), un modelo entrenado con datos reales de tráfico de red (LITNET-2020) para detectar intrusiones de manera efectiva. Aprovechando la transferencia de aprendizaje desde la arquitectura CANET, el modelo incorpora mejoras como capas de atención de una sola cabeza para capturar dependencias espacio-temporales, la función de pérdida Equalization Loss V2 (EQLv2) para abordar dinámicamente el desbalance de clases, y la codificación CatBoost para procesar eficientemente características categóricas de alta cardinalidad. Estas mejoras permiten al modelo manejar la complejidad de entornos de red del mundo real. Los resultados obtenidos demuestran un rendimiento superior en comparación con los métodos tradicionales de aprendizaje automático, alcanzando una tasa de detección, F1-score y precisión del 99.9559 %, con una tasa de falsos positivos (FPR) excepcionalmente baja de 0.003675 %. Estos hallazgos destacan la robustez y aplicabilidad de **H.A.L.C.CO.N** en aplicaciones modernas de ciberseguridad.

Palabras clave: Aprendizaje Profundo, Atención jerárquica, Ciberseguridad, Redes neuronales convolucionales, Sistema de detección de intrusos, Transfer Learning

Abstract

The rapid increase in internet users and connected devices has significantly raised cybersecurity risks, necessitating advanced intrusion detection systems (IDS). This study (In this work) presents **H.A.L.C.CO.N** (Hierarchical Attention-based Loss Equalization with CatBoost-enhanced Convolutional Neural Network), a model trained with real network traffic data (LITNET-2020) to detect intrusions effectively. Leveraging transfer learning from the CANET architecture, the model incorporates enhancements such as single-head attention layers to capture spatio-temporal dependencies, Equalization Loss V2 (EQLv2), to dynamically address class imbalance, and CatBoost encoding to effectively process high-cardinality categorical features. These enhancements enable the model to handle complexities of real-world networks environments. The results demonstrate superior performance compared to traditional machine learning methods, achieving a detection rate, F1-score, and accuracy of 99.9559 %, with an exceptionally low false positive rate (FPR) of 0.003675 %. These findings underscore the robustness and practicality of **H.A.L.C.CO.N** for modern cybersecurity applications.

Keywords: Convolutional neural networks, Cybersecurity, Deep Learning, Hierarchical Attention, Intrusion detection systems, Transfer Learning.

Capítulo 1

Generalidades

1.1. Introducción

El número de usuarios de Internet está en aumento en todo el mundo lo cual tiene un impacto directo en la cantidad de dispositivos conectados a Internet. Como consecuencia, los datos almacenados en nuestros dispositivos personales han aumentado de valor significativamente.

Las empresas, escuelas y organismos gubernamentales, cada vez están permitiendo a las personas trabajar y estudiar desde casa. Las redes son más vulnerables al robo y la perdida de información. Al mismo tiempo el acceso al Internet está ampliamente disponible y es barato pero también cualquier persona involucrada en los delitos cibernéticos puede lanzar un ataque de red sin importar donde están ubicados físicamente. Los ataques a las redes son intrusiones ilegales en redes privadas con la intención de dañar, destruir o robar información.

El alcance y la profundidad de las tecnologías de seguridad para computadoras y redes continúan evolucionando en respuesta a la naturaleza cambiante de las amenazas que enfrentan. Un sistema de detección de intrusos es una de las tecnologías más importantes que se están dando énfasis para combatir las amenazas. Los sistemas de detección de intrusos (SDI) son esenciales para cualquier arquitectura de seguridad integral. El objetivo general de un SDI es seguir detectando cualquier signo de comportamiento malicioso o riesgos previamente identificados en los datos de la red. Un SDI alerta al administrador de TI de una posible intrusión en la red después de una amenaza detectada.

Varios estudios demostraron que las técnicas tradicionales basadas en el aprendizaje au-

tomático, como k-Nearest Neighbors, Support Vector Machine y Naive Bayes Li W.; Yi P.; Wu Y.; Pan L.; Li, 2014; Wagh S.K.; Pachghare V.K.; Kolhe, 2013 se han aplicado al SDI. Sin embargo, los métodos tradicionales de aprendizaje automático no son adecuados para la detección de intrusiones en la red a gran escala con el enriquecimiento gradual de las categorías de ataques de red, debido a sus limitaciones para cambiar el aprendizaje de características. En los últimos años, los métodos basados en el aprendizaje profundo (DL) se están aplicando gradualmente a SDI.

En particular, varios estudios recientes Vinayakumar, Alazab, Soman, Poornachandran, Al-Nemrat y Venkatraman Vinayakumar et al., 2019; Yan, Jin, Lee y Liu Yan et al., 2018 han demostrado que las técnicas de aprendizaje profundo (DL) pueden lograr una mejora importante en la precisión en comparación con las técnicas convencionales de aprendizaje automático (ML). Aunque los métodos basados en el aprendizaje profundo existentes mejoran la tasa de detección (DR), debido al aprendizaje insuficiente de características, todavía sufren de una tasa de falsos positivos (FPR) alta.

Para abordar los desafíos anteriores en los SDI, Ren K. et Al Ren et al., 2023, propusieron un modelo llamado convolucional simple pero efectivo que incorpora características espacio temporales. Este modelo puede aprender automáticamente a centrarse en las características relevantes sin supervisión adicional. Al mismo tiempo, para resolver el problema del desequilibrio de clases, el modelo propone la pérdida de ecualización sensible al costo v2 (EQL v2) Tan et al., 2021 para ponderar las clases minoritarias.

Se realizaron varios experimentos donde se demuestra que CANET mejora constantemente la precisión de la predicción en diferentes conjuntos de datos y tamaños de entrenamiento al tiempo que logra un rendimiento de vanguardia sin requerir un preprocesamiento de datos adicional.

A pesar de la extensa investigación usando aprendizaje profundo, para hacer sistemas de detección de intrusos, la mayoría de los modelos han probado su efectividad usando datos sintéticos como NSL-KDD y UNSW-NB15 Moustafa y Slay, 2015 y es el mismo caso es para el modelo llamado CANET.

Ante la constante evolución de la inteligencia artificial y el cambio de escenarios, surge la transferencia de aprendizaje *(Transfer Learning TL)* que se ocupa de cómo los sistemas inteligentes pueden adaptarse rápidamente a nuevas situaciones, nuevas tareas y nuevos entornos. Proporciona a los sistemas de aprendizaje profundo la capacidad de aprovechar los datos y modelos

auxiliares para ayudar a resolver problemas de destino cuando solo hay una pequeña cantidad de datos disponibles en el dominio de destino. Esto hace que estos sistemas sean más fiables y robustos, evitando que un modelo de aprendizaje profundo se enfrenta a cambios imprevisibles se desvíe demasiado del rendimiento esperado. A nivel empresarial, el aprendizaje por transferencia permite reutilizar el conocimiento para que la experiencia adquirida una vez se pueda aplicar repetidamente al mundo real Qiang Yang y Pan, 2020.

1.2. Planteamiento del problema

1.2.1. Definición del problema

Los esfuerzos para automatizar la detección de intrusos, es un problema central en la ciberseguridad. Se trata de un campo de investigación muy dinámico que ha recibido mucha atención, pero los resultados obtenidos hasta ahora no se han reflejado en una automatización completa, practica y efectiva del trabajo de vigilancia Varga et al., 2023.

Sommer y Paxson Sommer y Paxson, 2010 propusieron explicaciones de por qué la investigación basada en enfoques de aprendizaje automático han tenido resultados deficientes:

1. El alto costo de los errores de clasificación (por ejemplo, la sensibilidad a los falsos positivos es demasiado alta debido a la gran cantidad de tráfico),
2. La variedad de formatos en los datos de entrada (por ejemplo, numerosos protocolos con diferentes frecuencia de aparición y contenido),
3. La falta de datos de entrenamiento (por ejemplo, tráfico de red realista, donde tanto los ataques y los flujos normales de tráfico se han etiquetado adecuadamente),
4. Datos de salida ambiguos (por ejemplo, preguntas sobre el significado exacto de una anomalía y cómo debería investigarse más a fondo),
5. Dificultades para determinar el valor o la viabilidad de las soluciones propuestas (en gran medida como efecto de los puntos 3 y 4).

La efectividad de SDI se evalúa en función de su desempeño para reconocer ataques, lo que requiere un conjunto de datos de red que proporcione ejemplos de tráfico de red normal y anormal Divekar et al., 2018.

Los antiguos conjuntos de datos de referencia como KDDCup'99 Siddique et al., 2019 y NSL-KDD Tavallaee et al., 2009 se han utilizado ampliamente para evaluar la precisión del reconocimiento de ataques de red Gao et al., 2019; Yao, Fu et al., 2019; Yao, Wang et al., 2019. Sin embargo, estos conjuntos de datos se han vuelto obsoletos debido al rápido desarrollo de las tecnologías de red y la aparición de nuevas amenazas de ciberseguridad y tipos de ataques de red Khraisat et al., 2019.

Muchos de los conjuntos de datos de referencia de flujo de red se crearon artificialmente, o los datos se recopilaron de un entorno altamente controlado, lo que los hace no representativos de los flujos de red de la vida real, mientras que los modelos entrenados en estos conjuntos de datos no pueden hacer frente a los ataques de red del mundo real Vasilomanolakis et al., 2016. Además, a medida que el tráfico de red de la vida real crece enormemente, el etiquetado manual de datos de conjuntos de datos cada vez mayores se convierte en una tarea inviable.

Para paliar este problema, Damasevicius et al, presentó LITNET-2020 et al, 2019, un nuevo conjunto de datos de referencia de redes obtenido de la red académica del mundo real. Este conjunto de datos contiene ejemplos reales de tráfico de red normal y bajo ataque también describe 85 características de flujo de red del conjunto de datos y 12 tipos de ataques. Este conjunto de datos de red está disponible de forma gratuita para fines de investigación.

Uno de los esfuerzos mas recientes para sistemas de detección de intrusos automatizado es el modelo llamado CANET Ren et al., 2023, una red neuronal convolucional con atención jerárquica que demostró sobresalir en el estado del arte en términos de precisión, relación de detección y relación de falsos positivos, pero el desempeño de este modelo fue evaluado con dos conjuntos de datos ampliamente usados NSL-KDD y UNSW-NB15 Moustafa y Slay, 2015.

Aunque que modelo CANET sea de alta calidad usando datos sintéticos, también puede cometer errores, especialmente cuando el modelo se aplica a escenarios diferentes de sus entornos de entrenamiento. La capacidad de detección del sistema puede disminuir drásticamente. Esto se debe a que el modelo entrenado se aplica a un escenario "diferente". Esta caída en el rendimiento muestra que los modelos pueden estar desactualizados y necesitan actualizarse cuando se

producen nuevas situaciones.

Es esta necesidad de actualizar o transferir modelos de un escenario a otro lo que le da importancia de aplicar metodología la transferencia de aprendizaje al modelo CANET.

Utilizando el modelo CANET como referencia, se usará la metodología de transferencia de aprendizaje para generar un nuevo modelo de red neuronal convolucional con atención jerárquica que utilice datos LIT-NET 2020 et al, 2019 para demostrar que la efectividad del sistema de detección de intrusos usando datos reales es competitiva en comparación a los datos sintéticos.

1.2.2. Delimitación de la investigación

- Esta investigación se visualiza la creación de un modelo red neuronal convolucional de atención jerárquica para datos reales, mediante la metodología transferencia de aprendizaje basado en el modelo CANET, que servirá para una posterior implementación de un sistema detección de intrusos automático.
- La efectividad del modelo propuesto será evaluado usando un conjunto de datos de flujo de red real llamado LITNET.
- El trabajo se limitará al uso de datos LITNET, ya que están disponibles para investigación de forma gratuita y servirán para la evaluación de la efectividad del modelo creado.
- Este trabajo no presenta limitaciones éticas o derechos de autor ya que tanto el modelo de referencia CANET y el conjunto de datos LITNET, esta disponibles de forma gratuita.
- En caso de que se presenten limitaciones no previstas en este trabajo, se puede considerar evaluar el comportamiento del dato real en modelos ya existentes.

Se obtendrá un modelo de red neuronal convolucional de atención jerárquica, para la detección de intrusos que podrá ser implementado en sistemas de redes de datos reales.

1.3. Preguntas de investigación e hipótesis

Este trabajo de tesis busca contestar las siguientes preguntas:

- ¿Es posible construir un nuevo modelo de red neuronal artificial usando la metodología de transferencia de aprendizaje?
- ¿Es posible procesar los datos reales de red para adaptarlos al nuevo modelo?
- ¿Se mantendrá la efectividad obtenida de CANET en el nuevo modelo construido con transferencia de aprendizaje usando datos reales?

La hipótesis de la tesis es que el modelo obtenido con la metodología de transferencia de aprendizaje y evaluado con un conjunto datos reales de trafico de red LIT-NET 2020, obtendrá una precisión 85 %, tasa de detección 95 % y tasa de falsos positivos 0.8 %.

1.4. Objetivo general

Construir un modelo de red neuronal convolucional de atención jerárquica, para la detección de intrusos usando datos de flujo de red real, mediante la metodología transferencia de aprendizaje.

1.5. Objetivos específicos

- Construir un modelo que utilice datos reales de flujo de red, mediante la metodología de transferencia de aprendizaje.
- Llevar a cabo una ingeniera de clases usando metodología de transferencia de aprendizaje para la adecuación los datos reales al modelo.
- Verificar la efectividad el SDI propuesto usando la métricas de precisión, tasa de detección y tasa de falsos positivos.

1.6. Justificación

En la industria de la ciberseguridad, los sistemas de detección de intrusos automáticos son un campo investigación que tienen mucho esfuerzo y atención internacional. Y se requieren de resultados que sean efectivos, prácticos y completos.

La efectividad de estos sistemas se evalúan usando métricas que miden su capacidad para reconocer ataques por lo tanto se requieren de conjuntos de datos de red que proporcione ejemplos de tráfico de red normal y anormal Divekar et al., 2018.

Sommer y Paxson Sommer y Paxson, 2010 señalan que una de las limitantes para que los sistemas de detección de intrusos basados en aprendizaje automáticos brinden resultados prácticos es la falta de datos tráfico de red realista, donde tanto los ataques y los flujos normales de tráfico se han etiquetado adecuadamente.

El uso del conjunto de datos LITNET se justifica por que es conjunto de datos de referencia de redes obtenido de la red académica del mundo real y ademas etiquetado adecuadamente. El modelo CANET usado como referencia en este trabajo, se justifica porque es simple pero efectivo ademas incorpora características espacio temporales. Este modelo puede aprender automáticamente a centrarse en las características relevantes sin supervisión adicional. Al mismo tiempo, para resolver el problema del desequilibrio de clases, el modelo propone el método de la pérdida de ecualización sensible al costo v2 (EQL v2) Tan et al., 2021.

En la transferencia, el aprendizaje permite reutilizar el conocimiento, de modo que la experiencia adquirida una vez pueda aplicarse repetidamente al mundo real. Si aplicamos el aprendizaje por transferencia en este trabajo de investigación, podemos obtener un sistema de aprendizaje automático de por vida que puede extraer conocimiento de una sucesión de experiencias de resolución de problemas, tanto en un largo período de tiempo como de una gran variedad de tareas. El aprendizaje por transferencia dotará al sistema inteligente la capacidad de aprendizaje permanente Qiang Yang y Pan, 2020.

1.7. Metodología utilizada

Dada la naturaleza de esta investigación se utilizara el método cuantitativo adaptado del llamado CRISP-DM *(CRoss-Industry Standard Process for Data Mining)*, que incluye los siguientes pasos.

1. Planteamiento del problema desde una visión global.
    - Determinar los objetivos.
    - Determinar posible soluciones.
    - Verificar si existen soluciones similares.
    - Decidir las aproximaciones del problema
    - Definir las métricas para evaluar el desempeño
    - Definir Hipótesis
2. Obtener los datos. Aquí se en listan los puntos esenciales para obtener los datos.
    - Listar los datos disponibles y estimar cuántos serán necesarios
    - Documentar cómo y dónde encontrar los datos
    - Comprobar cuánto espacio ocuparán los datos.
    - Comprobar si existen limitaciones en el uso de los datos, obteniendo autorización si es necesario.
    - Convertir los datos a un formato en el que se puedan manipular fácilmente
    - Asegurar de que cualquier información sensible es eliminada o protegida.
    - Comprobar el tipo de los datos
    - Generar datos de prueba
    - Automatizar el máximo numero de puntos en el proceso para obtener nuevos datos de manera regular, en caso de ser posible.
3. Explorar y visualizar los datos. En esta fase del desarrollo de la investigación el objetivo es explorar y visualizar los datos. Para ello se seguirán los siguientes pasos: .

- Crear una copia del conjunto de datos para explorarlo (se utilizará una pequeña muestra si es necesario).
- Crear un notebook para la exploración de los datos (lo que se conoce como EDA, o exploratory data analysis en inglés).
- Analizar cada atributo y sus características: nombre, tipo (categórico o numérico), cantidad de valores inexistentes (ruido en los datos, usabilidad para la tarea en cuestión).
- Identificar las características objetivo, el target o ground truth.
- Visualizar los datos.
- Estudiar correlaciones entre características.
- Crear algoritmo
- Identificar posibles transformaciones que pueden aplicar a los datos para obtener mejores resultados.
- Investigar si existen datos adicionales que se puedan usar (y vuelve al punto anterior).
- Documentar todos los descubrimientos y lo resultados.

4. Transferencia de aprendizaje y Preparar los datos para el modelo, en esta fase los datos se preparan para que el modelo de red neuronal se capaz de usarlos para entrenar. Se seguirán los siguientes pasos:

   - Trabajar con copias del conjunto de datos, dejar el original intacto.
   - Implementar funciones para todas las transformaciones que se apliquen, de esta manera se podrá aplicar a nuevos datos o en futuros investigaciones.
   - Limpiar los datos eliminando anomalías y rellenando (o eliminando) los valores que falten.
   - Eligir aquellos atributos con los que se trabajara y eliminar el resto.
   - Aplicar Ingeniera de clases: separar las características en numéricas y categóricas, procesar los datos que sean necesarios (texto, fechas, ...), añadir nuevas características, según aplique.
   - Normalizar los datos.

- Definir lo que se va a transferir.
- Definir cuando se va transferir.
- Definir como se va transferir el aprendizaje.

5. Entrenar modelo. En esta fase se seguirán los siguientes pasos:
   - Probar el modelo con hiper parámetros iniciales.
   - Comparar métricas obtenidas entre los diferentes hiper parámetros
   - Analizar los errores de los diferentes modelos
   - Iterar el proceso con diferentes combinaciones de clases.
   - Seleccionar los modelos mas prometedores, de preferencia los que cometen diferentes errores.
6. Ajustar modelo (optimización de hiper parámetros). en este paso se usaran todos los datos disponibles con la optimización de hiper parámetros.
7. Presentar resultados. para esta fase se realizaran las siguientes actividades:
   - Documentar todo el trabajo que has llevado a cabo.
   - Crear una presentación con visualizaciones llamativas e informativas que reflejen la visión global y transmitan las conclusiones principales.
   - Explicar la solución y porque has logrado los objetivos.
   - Describir lo aprendido en el proceso, las cosas que han funcionado y las que no, las hipótesis asumidas y las limitaciones del sistema.
   - Generar, someter a revisión y publicar articulo científico.
   - Entregar Tesis.
   - Exponer resultados en congresos de relevancia nacional o internacional.

Capítulo 2

Marco teórico

2.1. Conceptos y teorías fundamentales de la investigación

Para este trabajo de investigación es necesario profundizar conceptos como la inteligencia artificial, aprendizaje automático, aprendizaje profundo, ciberseguridad y la transferencia de aprendizaje.

2.1.1. Inteligencia artificial

El término inteligencia artificial se introdujo por primera vez como disciplina académica en 1956 en una conferencia en Dartmouth Ahmad et al., 2018 y, aunque al principio pareció interesarse solo en la programación relacionada con estrategias de damas en la década de 1960 y otros temas similares, finalmente ganó el interés del Departamento de Defensa de los Estados Unidos (DoD) que invirtió en la tecnología Anyoha, 2017 y, poco a poco, el aprendizaje automático y la inteligencia artificial evolucionaron hasta convertirse en una tecnología viable para la resolución de problemas y muchas otras cosas.

En la actualidad, la IA ya se ha afianzado con fuerza en grandes sectores industriales como la defensa, exploración espacial, salud y la automoción, y también se está viendo una comportamiento similar en el sector manufacturero, como la Automatización Robótica de Procesos (RPA) y otras empresas de automatización.

Según Markets and Markets, el tamaño del mercado de inteligencia artificial está valorado en

USD $ 150.2 mil millones en 2023 y se estima que se expandirá a una tasa de crecimiento anual de casi el 36.8 % hasta 2030 con la IA basada en datos, el aprendizaje profundo y la necesidad de lograr la autonomía robótica que liderarán el camino Marketsandmarkets, 2023.

Originalmente, la Inteligencia Artificial (IA) se introdujo como un concepto para imitar el cerebro humano e investigar los problemas del mundo real con un enfoque humano holístico. Se puede considerar como una combinación de tecnología de la información e inteligencia fisiológica, que se puede utilizar computacional mente para alcanzar objetivos Vähäkainu y Lehto, 2023.

La inteligencia artificial puede realizar varias tareas utilizando métodos, como el procesamiento del lenguaje natural (NLP), el análisis predictivo y prescriptivo o el aprendizaje automático y profundo. Estos métodos también se pueden utilizar para resolver problemas de ciberseguridad. «What's the difference between Artificial Intelligence, Machine Learning and Deep Learning?», 2019

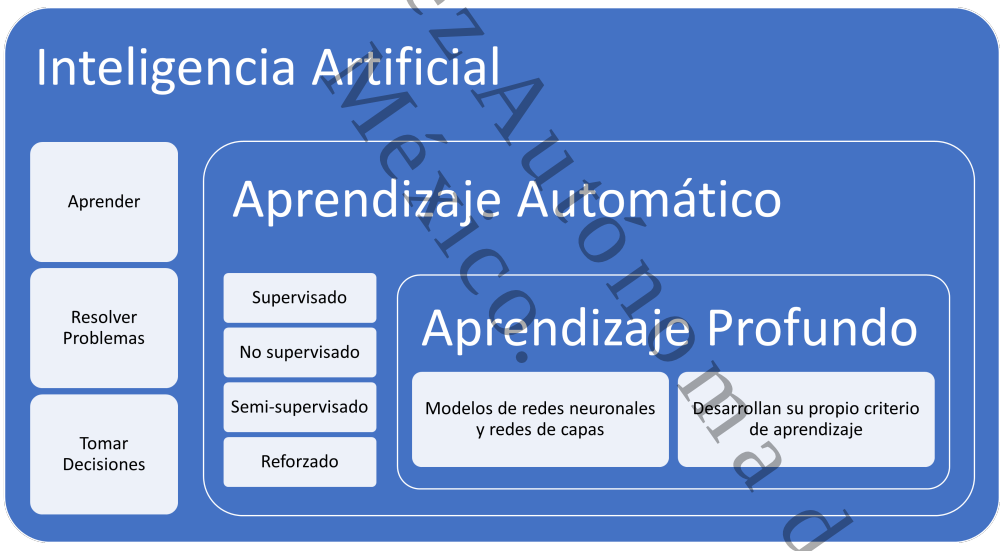


Figura 2.1. Inteligencia Artificial

2.1.2. Aprendizaje automático

El aprendizaje automático se puede dividir en dos tipos principales, que son el aprendizaje automático supervisado y el aprendizaje automático no supervisado.

El aprendizaje automático supervisado utiliza algoritmos que pueden aprender de los datos introducidos por un usuario; por lo tanto, se requiere la intervención humana para etiquetar, clasi-

ficar e ingresar los datos en el algoritmo Estapé, 2021.

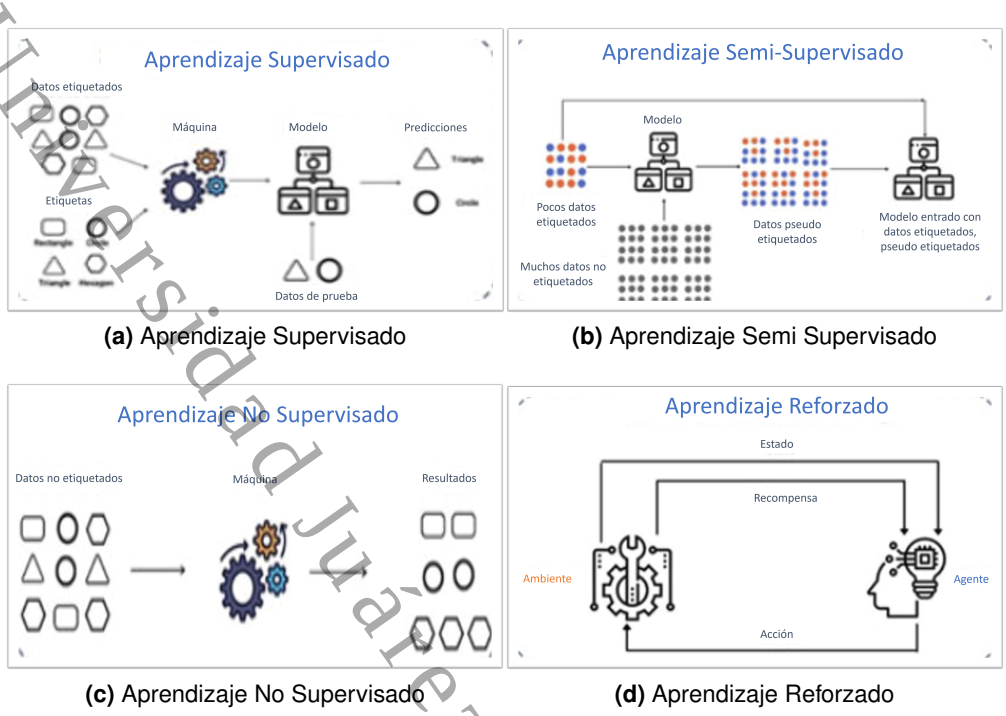
El aprendizaje automático supervisado se puede agrupar en problemas de regresión y clasificación. Los problemas de clasificación utilizan un algoritmo específico para asignar los datos de prueba en categorías y luego clasificar un objeto dentro de diferentes clases, por ejemplo, determinar si un correo electrónico es spam y separar los correos electrónicos no deseados de la bandeja de entrada. El método de regresión predice un valor numérico, por ejemplo, al comprobar la demanda de ocupación de un hotel. Los métodos de clasificación más conocidos son, por ejemplo, las máquinas de vectores de soporte (SVM), los métodos Naïve Bayes y K-Nearest neighbor. Los métodos de regresión típicamente utilizados son la regresión lineal, SVR, los métodos de conjunto, los árboles de decisión y las redes neuronales populares SparkCognition, 2018.

El aprendizaje automático no supervisado es el entrenamiento de una máquina utilizando información que no está clasificada ni etiquetada. El método no supervisado utiliza algoritmos específicos de aprendizaje automático para analizar y agrupar conjuntos de datos sin etiquetar sin necesidad de intervención humana. El algoritmo de ML agrupa la información no clasificada de acuerdo con diferencias, patrones y similitudes sin entrenamiento previo de datos al encontrar una estructura oculta en datos no etiquetados.

Los modelos no supervisados se pueden agrupar en dos tipos principales de la siguiente manera: agrupamiento y asociación. La agrupación en clústeres es una técnica de minería de datos, que se puede utilizar para clasificar datos sin etiquetar en grupos en función de sus similitudes o diferencias. La agrupación en clústeres se puede utilizar, por ejemplo, para realizar la segmentación de clientes de acuerdo con lo que han comprado, por ejemplo, mediante el uso de agrupaciones K-means para asignar puntos de datos similares en grupos. La asociación puede descubrir reglas y encontrar relaciones entre variables dentro del conjunto de datos específico. El método se ha utilizado, por ejemplo, para el análisis de la carros de compra y los motores de recomendación.

2.1.3. Aprendizaje profundo

El aprendizaje automático tiene un subcampo, el aprendizaje profundo, donde el aprendizaje se realiza con modelos que tienen múltiples capas dentro de su estructura. La enseñanza a



(a) Aprendizaje Supervisado **(b)** Aprendizaje Semi Supervisado

(c) Aprendizaje No Supervisado **(d)** Aprendizaje Reforzado

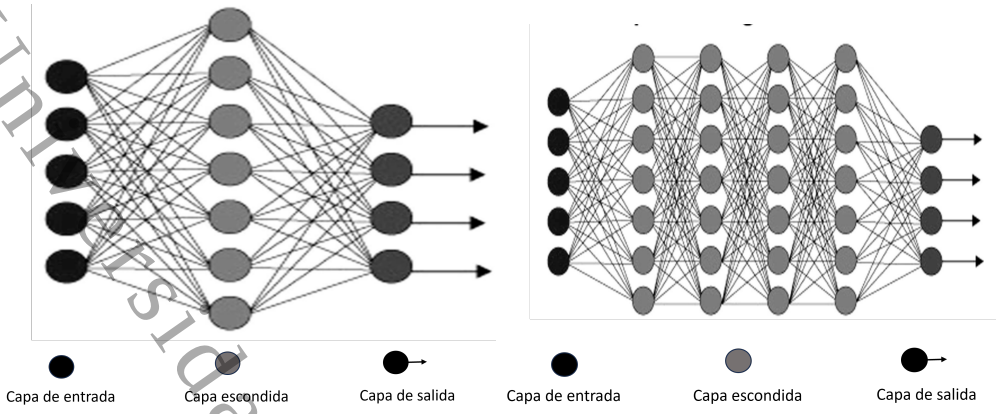
Figura 2.2. Aprendizaje automático

distancia puede ser supervisada, semisupervisada o no supervisada. La profundidad adicional puede ayudar a los modelos a aprender asociaciones más complejas dentro de los datos dados que los modelos regulares de IALeCun et al., 2015; por lo tanto, los modelos DL se denominan profundos.

La red neuronal (NN) es el modelo base y la columna vertebral de los algoritmos de aprendizaje profundo, que se utiliza en el desarrollo de soluciones de inteligencia artificial. El aprendizaje profundo es una colección de nodos estructurados e interconectados cuyos valores están compuestos por todos los pesos de las conexiones que llegan a cada nodo. Cada valor de un nodo se introduce en una función de activación, como una unidad lineal rectificada (ReLu). La función de activación suele ser la misma para todos los nodos de la misma capa.

2.1.4. Transferencia de aprendizaje

El aprendizaje por transferencia puede hacer que los sistemas de IA y aprendizaje profundo sean más fiables y sólidos. A menudo se da el caso de que, al construir un modelo de aprendizaje profundo, no se pueden prever todas las situaciones futuras. El aprendizaje por transferencia lleva



(a) Red Neuronal **(b)** Aprendizaje Profundo

este enfoque más allá, al permitir que el modelo sea complejo y al mismo tiempo esté preparado para los cambios cuando realmente se produzcan. Además, cuando se enfrentan a cambios imprevisibles y se lleva un modelo aprendido a través de los límites del dominio, el aprendizaje por transferencia se asegura de que el rendimiento del modelo no se desvíe demasiado del rendimiento esperado Qiang Yang y Pan, 2020.

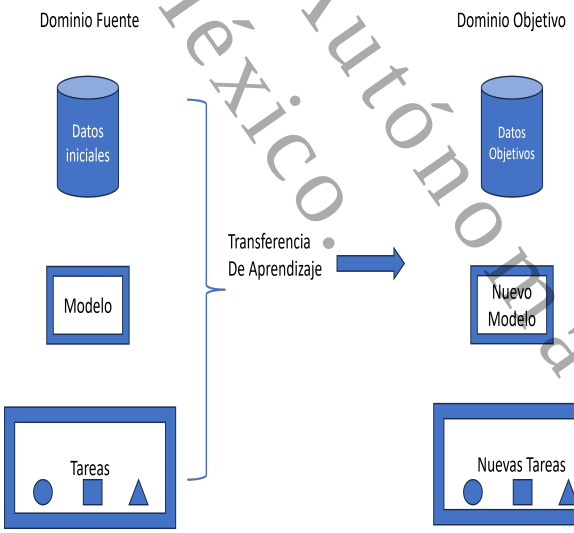


Figura 2.3. Una ilustración del proceso de transferencia de aprendizaje *(Transfer Learning TL)*

2.1.5. Ciberseguridad

La ciberseguridad se puede definir como un conjunto de acciones tomadas en defensa contra los ciberataques y sus consecuencias, e incluye la implementación de las contra medidas necesarias. La ciberseguridad se basa en el análisis de amenazas de una organización o institución. La

estructura y los elementos de la estrategia de ciberseguridad de una organización y su programa de implementación se basan en las amenazas estimadas y los análisis de riesgos. En muchos casos, es necesario preparar varias estrategias y directrices de ciberseguridad específicas para una organización Lehto, 2015.

Los temas clave de investigación en esta disciplina incluyen técnicas asociadas con la detección de diferentes anomalías de comportamiento de red y malware.

1. Identificar.
2. Proteger.
3. Detectar.
4. Responder.
5. Recuperar.

Figura 2.4. Marco de ciberseguridad NIST«NIST, cybersecurity framework», s.f.

2.1.6. Automatización en el ciberseguridad

Por automatización, nos referimos aquí a la ejecución de alguna tarea por parte de un agente máquina (generalmente una computadora) que previamente era llevada a cabo por un humano, de acuerdo con la definición de Parasuraman y Riley Parasuraman y Riley, 1997.

Las soluciones automatizadas han demostrado un gran potencial dentro del campo de la ciberseguridad, y gran parte de la investigación actual se dirige hacia soluciones automatizadas.

Un ejemplos de agente máquina en ciberseguridad son los sistemas de detección de intrusos y las herramientas de evaluación de vulnerabilidades. Uno de los objetivos de las investigaciones actuales en ciberseguridad es automatizar en la mayor medida posible, las actividades de la ciberseguridad, pero a menudo existen obstáculos técnicos que impiden resultados exitosos de los esfuerzos de automatización.

Las ciberamenazas avanzadas de hoy en día son elaboradas, sostenibles y explotan todas las oportunidades posibles en un objetivo. Además, se han vuelto asequibles de implementar con herramientas nuevas y avanzadas disponibles para todos Li, 2018. Los atacantes ahora despliegan y usan los modelos y técnicas de Inteligencia Artificial ofensivos (o manipulan los defensivos) en su beneficio. En cualquier caso, un uso exitoso de la Inteligencia Artificial en ciberseguridad o cualquier otro campo comparable requiere datos de entrenamiento de alta calidad, recursos suficientes para la creación de modelos y algoritmos efectivos Hoffman, 2021.

Dado el aumento exponencial de incidentes en la ciberseguridad se debe hacer uso de los avances recientes del aprendizaje profundo para automatizar las actividades involucradas en esta disciplina.

2.1.7. Netflow

Es importante comenzar con remapear y entender la informacion del flujo de red LITNET, el cual utiliza un protocolo llamada NetFLow V9.

NetFlow es un protocolo desarrollado por Cisco Systems que se utiliza para recopilar información de tráfico IP a medida que fluye dentro y a través de una interfaz de red. Esta tecnología permite a los administradores de red y a los dispositivos de seguridad obtener información detallada sobre el tráfico de red, incluyendo el origen, destino, volumen, y tipo de tráfico, entre otros datos. NetFlow se ha convertido en un estándar de facto para la monitorización del tráfico de red, y su uso se extiende a la planificación de la capacidad de la red, análisis de seguridad, monitoreo del rendimiento de la red y diversas aplicaciones de análisis de tráfico.

En la siguiente tabla se muestra el diccionario de entrada para leer los datos LITNET y comprenderlos.

2.2. Estado del arte en detección de intrusos

2.2.1. Aprendizaje profundo en ciberseguridad

La automatización en la detección de intrusos se ha convertido en una prioridad ante la creciente sofisticación de las amenazas cibernéticas. El uso de técnicas de aprendizaje automático y profundo ha permitido construir sistemas capaces de procesar grandes volúmenes de datos de tráfico en tiempo real, identificar patrones complejos y responder de forma proactiva. Dentro de este contexto, tres líneas de investigación destacan por su impacto reciente: el uso de Redes Neuronales Convolucionales (CNN), los mecanismos de atención jerárquica, y el aprendizaje por transferencia.

2.2.2. Redes Neuronales Convolucionales (CNN)

Las CNN han demostrado gran eficacia en tareas de visión por computadora, y han sido adaptadas exitosamente para analizar secuencias de tráfico de red en sistemas de detección

de intrusos (IDS). Su capacidad para aprender jerarquías de características a partir de datos de entrada las hace particularmente adecuadas para reconocer patrones anómalos y sutiles en el comportamiento de red. Estudios como los de Kim et al. Kim et al., 2016, Javaid et al. Javaid et al., 2018 y Yin et al. Yin et al., 2017 evidencian su efectividad en la detección de ataques como DDoS, intrusiones por malware, y escaneos de puertos.

Sin embargo, las CNN enfrentan desafíos importantes. Requieren grandes volúmenes de datos etiquetados para entrenarse correctamente, lo cual limita su aplicabilidad en entornos reales debido a restricciones de privacidad. Además, su capacidad de adaptación a ataques emergentes sigue siendo un tema de investigación activa. Para enfrentar estos retos, se han propuesto enfoques híbridos que combinan CNN con técnicas de atención y transferencia de conocimiento.

2.2.3. Mecanismos de atención jerárquica

Inspirados en la capacidad cognitiva humana de enfocar recursos en información relevante, los mecanismos de atención —particularmente en su forma jerárquica— han sido incorporados en modelos de detección de intrusos para mejorar la precisión y reducir los falsos positivos. A través de la ponderación diferencial de componentes de entrada, estos mecanismos permiten identificar características clave del tráfico de red sin distraerse con ruido irrelevante.

Vaswani et al. Vaswani et al., 2017 introdujeron la atención como parte fundamental del modelo Transformer. En el ámbito de la ciberseguridad, investigaciones como las de Zhou et al. Zhou et al., 2016 y Lin et al. Lin et al., 2017 han mostrado que la atención jerárquica permite detectar señales maliciosas sutiles en secuencias de datos, contribuyendo a una clasificación más robusta y explicable.

2.2.4. Aprendizaje por transferencia

El aprendizaje por transferencia se basa en reutilizar conocimientos adquiridos en una tarea para mejorar el rendimiento en una tarea relacionada, lo cual resulta ventajoso cuando se dispone de pocos datos etiquetados. Pan y Yang Pan y Yang, 2010 clasifican diversas técnicas de transferencia y destacan su aplicabilidad cuando los dominios fuente y objetivo presentan ciertas similitudes estructurales.

En la detección de intrusos, esta técnica permite utilizar modelos preentrenados en grandes datasets, como los de visión por computadora, y adaptarlos al análisis de tráfico de red. Alrawashdeh y Purdy Alrawashdeh y Purdy, 2016 demostraron que esta estrategia puede incrementar significativamente la precisión sin necesidad de entrenar modelos desde cero. No obstante, la diferencia de dominios (por ejemplo, imágenes vs. datos de red) puede limitar la efectividad de la transferencia directa, razón por la cual se han desarrollado métodos de adaptación de dominio y ajustes finos que permiten generalizar mejor en entornos cibernéticos.

2.2.5. Automatización y adaptabilidad en la detección

La automatización total de la detección de intrusos representa la convergencia de las técnicas anteriores. El desarrollo de modelos que combinen CNN, atención jerárquica y aprendizaje por transferencia busca alcanzar una detección precisa, explicable y de bajo costo computacional. Los desafíos persisten, particularmente en la adaptabilidad a nuevas amenazas (concept drift), la reducción de falsos positivos/negativos y la integración en sistemas de producción en tiempo real.

Tendencias emergentes incluyen el uso de arquitecturas multimodales, el aprendizaje continuo, y la automatización de respuestas a incidentes. La literatura sugiere que los modelos capaces de integrar estas capacidades —como es el caso de H.AL.C.C.O.N— están mejor posicionados para responder a los retos dinámicos del entorno cibernético moderno.

2.2.6. Automatización y aprendizaje profundo en la detección de intrusos

La ciberseguridad es un campo dinámico que enfrenta amenazas cada vez más sofisticadas. En este contexto, la automatización de la detección de intrusos se ha vuelto fundamental para mejorar la eficacia y velocidad de respuesta ante ataques. Tecnologías como el aprendizaje automático (ML) y el aprendizaje profundo (DL) han permitido desarrollar sistemas capaces de analizar grandes volúmenes de tráfico de red en tiempo real, identificar patrones anómalos y activar mecanismos de defensa de manera proactiva.

Dentro de estas tecnologías, las Redes Neuronales Convolucionales (CNN) han demostrado ser particularmente efectivas para capturar características espacio-temporales del tráfico de red. Trabajos pioneros como los de Kim et al. Kim et al., 2016 y Javaid et al. Javaid et al., 2018

validaron su capacidad para detectar comportamientos maliciosos con alta precisión, destacando su potencial para adaptarse a nuevas amenazas sin supervisión intensiva.

Recientemente, se ha incorporado el uso de mecanismos de atención y aprendizaje por transferencia en modelos de detección de intrusos. Estos enfoques permiten a los modelos enfocarse en características relevantes y reutilizar estructuras previamente aprendidas en dominios relacionados, reduciendo el tiempo de entrenamiento y mejorando la generalización. No obstante, persisten desafíos importantes, como la adaptación a amenazas emergentes, el manejo del desbalance de clases y la minimización simultánea de falsos positivos y negativos.

Las tendencias actuales apuntan hacia sistemas más flexibles, capaces de operar en entornos dinámicos y con respuestas automatizadas. La exploración de arquitecturas híbridas, la integración de modalidades múltiples de datos (multimodalidad) y el desarrollo de estrategias de autoactualización representan líneas activas de investigación. En este panorama, el modelo **H.AL.C.C.O.N** se posiciona como una propuesta relevante al combinar atención jerárquica, codificación avanzada y transferencia de arquitectura sobre datos reales, contribuyendo a la evolución de sistemas de detección de intrusos adaptativos, eficientes y reproducibles.

2.3. Marco tecnológico

La construcción de un modelo de red neuronal convolucional (CNN) con atención jerárquica, utilizando aprendizaje por transferencia para un sistema de detección de intrusos y evaluado con datos reales de flujo de red, implica un marco tecnológico sofisticado. Este marco debe abordar desde la selección del conjunto de datos y la preprocesamiento hasta la arquitectura del modelo, el entrenamiento y la evaluación. Este marco tecnológico proporciona una base sólida para el desarrollo, entrenamiento, evaluación y despliegue de modelos avanzados de detección de intrusos utilizando CNN con atención jerárquica y aprendizaje por transferencia, garantizando que el sistema sea escalable, seguro y eficaz.

El desarrollo de este modelo requiere de hardware, software y herramientas adecuadas para facilitar la investigación, el desarrollo y la implementación. A continuación, se detalla un marco tecnológico integral:

2.3.1. Hardware

- Unidades de Procesamiento: Utilizar GPUs (Unidades de Procesamiento Gráfico) de alta capacidad, como las de la serie NVIDIA RTX, para entrenamiento y ejecución de modelos. Las GPUs son esenciales para acelerar las operaciones de las redes neuronales
- Estaciones de trabajo con CPUs multicore (por ejemplo, AMD Ryzen Threadripper o Intel Xeon) para gestionar el flujo de datos y almacenamiento de alta velocidad (SSDs NVMe) para almacenar grandes volúmenes de datos de flujo de red y modelos entrenados

2.3.2. Software y herramientas de desarrollo

- Sistema Operativo: Windows 11 por su robustez, seguridad y compatibilidad con herramientas de aprendizaje profundo.
- Entornos de Desarrollo y Bibliotecas:
- Python: Lenguaje de programación preferido por su simplicidad y las extensas bibliotecas disponibles para aprendizaje profundo.
- PyTorch: Biblioteca de aprendizaje profundo que soporta la creación, entrenamiento y despliegue de modelos de CNN con mecanismos de atención y ademas ofrece soporte para aprendizaje por transferencia.
- Keras: Interfaz de alto nivel para Pytorch, facilitando el diseño y entrenamiento de redes complejas.
- Hugging Face Transformers: Para implementar fácilmente mecanismos de atención y modelos preentrenados.
- TensorBoard o Visdom: Para visualizar de forma dinámica el proceso de entrenamiento, incluyendo pérdidas y precisión.
- Matplotlib y Seaborn: Para análisis estadístico y visualización de datos.

2.3.3. Herramientas de preprocesamiento de datos

- Pandas y NumPy: Para manipulación y operaciones con datos.
- Scikit-learn: Para preprocesamiento de datos, como normalización y codificación de variables categóricas.

2.3.4. Plataformas y servicios en la nube

Microsoft Azure: Ofrece infraestructura escalable (GPUs, almacenamiento) y servicios para entrenamiento de modelos y despliegue de aplicaciones de detección de intrusos.

Capítulo 3

Metodología

3.1. Obtención de datos

3.1.1. Descripción del conjunto de datos LITNET

En esta parte, se describe el entorno de red utilizado para la recopilación de datos de tráfico de red, la descripción de los ataques de red que están presentes en el conjunto de datos, las características descriptivas del conjunto de datos y la disponibilidad del conjunto de datos.

3.1.1.1. Entorno de red

La infraestructura consiste en la conexión de nodos con equipos operativos y redes de comunicación que conectan a esos exportadores (CAPACITY, CYTI1, KTU UNIVERSITY 1, KTU UNIVERSITY 2 y FIREWALL). La topología LITNET NetFlow consta de dos partes principales (emisores y recopilador). Los remitentes de NetFlow son routers Cisco (Cisco Systems Inc., San José, CA, EE. UU.). Los firewalls de ultima generación de alto rendimiento Fortige (FG-1500D) que analizan los datos que han pasado a través de él, procesan los datos y los envia a uno o más recopiladores de servidores NetFlow. El servidor NetFlow (recopilador) es un servidor con el software adecuado (nfcapd, nfdump, nfexpire, nfprofile, nfreplay y nftrack. Versión: 1.6.15), que se encarga de recibir, almacenar y filtrar datos. Se utiliza 4.9.0-11 amd64 1 SMP Debian 4.9.189 3+deb9u2 x86 64 sistema operativo GNU/Linux; procesador de CPU Intel Xeon de 4 núcleos (Skylake, IBRS); 10GB para el sistema y discos duros de 30T para la recogida de datos; y 8 GB

de RAM.

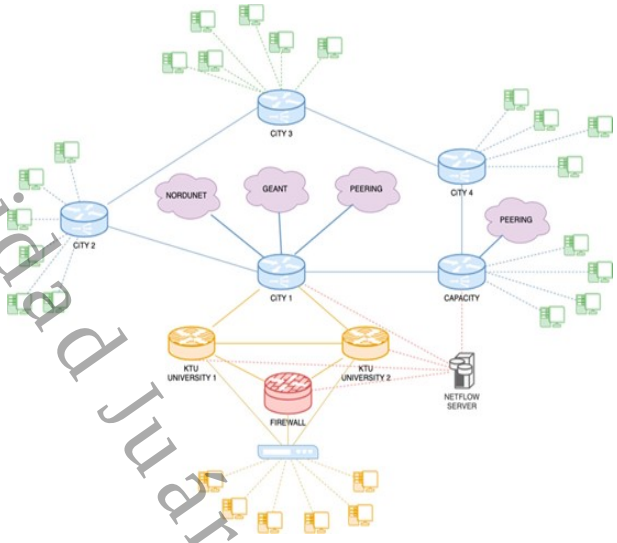


Figura 3.1. Infrraestructura LITNETet al, 2019

Cada uno de los exportadores de NetFlow (CAPACITY, CYTI1, KTU UNIVERSITY 1, KTU UNIVERSITY 2, y FIREWALL) monitorea continuamente los flujos que pasan a través de él (esta es una secuencia de paquetes de datos anteriores en una dirección desde un remitente específico a un destinatario específico) y los almacena en caché cuando recibe nuevo tráfico.

El anillo principal conecta las cinco universidades más grandes de Lituania, que son las siguientes: (CITY1) Universidad de Kaunas-Vytautas Magnus y Universidad Tecnológica de Kaunas, que es administradora de la Red Lituana de Investigación y Educación (LITNET) y de los nodos de conexión de mantenimiento y desarrollo y la Universidad de Vilna (CAPACITY); Universidad Técnica Gediminas de Vilnius (CAPACITY); Universidad de Klaipeda (CITY2); Universidad de Siauliai (CITY3); y la Facultad de Tecnologías y Negocios de KTU Panevezys (CITY4).

Por razones de eficiencia, el remitente de NetFlow solo escanea el primer paquete de la nueva secuencia, que guarda los valores correspondientes, y luego los paquetes posteriores de la misma secuencia se procesan de acuerdo con la misma directriz, lo que reduce la carga en el dispositivo de red.

La Universidad Tecnológica de Kaunas (FIREWALL) cuenta con una infraestructura de alta disponibilidad y un perímetro Fortigate 1500D con sistema operativo FortiOS, red de 80 Gbps, IPS (sistema de prevención de intrusiones) de 13 Gbps, NGFW (firewall de próxima generación) de 7 Gbps, y firewall de protección contra amenazas de 5 Gbps.
CITY1 tiene una salida a las redes de banda ancha NORDUNET y GEANT. CITY1 y CITY2 tienen una conexión de emparejamiento, y este es un proceso mediante el cual dos redes de Internet se conectan e intercambian tráfico.

CITY1 y CITY2 pueden transferir tráfico de datos directamente entre los usuarios de LITNET de cada uno. Todas las demás ciudades (CITY2, CITY3 y CITY4) tienen usuarios finales. Se trata de escuelas, municipios, otras organizaciones.

3.1.1.2. Descripción de los ataques de red contenidos en LITNET

A continuación se describen los tipos de ataque en el conjunto de datos propuesto.

El ataque Smurf sigue enviando las solicitudes de difusión del Protocolo de mensajes de control de Internet (ICMP) a la red en nombre del nodo de destino, con el objetivo de inundar el nodo con tráfico de red para ralentizar el nodo de destino.

Un ataque de inundación del Protocolo de mensajes de control de Internet (ICMP) es un ataque DoS que tiene como objetivo saturar un nodo de red objetivo con solicitudes de eco (pings) ICMP.
El ataque de inundación UDP es un ataque DoS que utiliza el protocolo de datagramas de usuario (UDP). Un ataque de inundación de DNS (DNS Flooding) es una variante específica de la aplicación de una inundación UDP, que se caracteriza por el envío de paquetes de red a cualquier dirección IP, utilizando el protocolo UDP y el puerto 53 como objetivo.

El ataque TCP SYN-flood es un ataque DoS distribuido (DDoS) que hace un mal uso de una parte del protocolo de enlace tridireccional ordinario del protocolo de control de transmisión (TCP) para drenar los recursos del nodo víctima y hacer que no responda. Los paquetes de paquetes

de ataque tienen indicadores S pero no tienen los indicadores AFRPU.

El ataque de inundación HTTP es un ataque DDoS, que explota las solicitudes GET o POST aparentemente legítimas del Protocolo de transferencia de hipertexto (HTTP) para atacar un servidor web o una aplicación. En un ataque complejo de capa 7, las inundaciones HTTP no utilizan paquetes mal formados, suplantación de identidad o reflexión y necesitan menos ancho de banda que otros tipos de ataques, con el fin de deshabilitar el servidor o sitio de la víctima. Los paquetes de ataque se dirigen solo al puerto 80.

El ataque LAND es un ataque DoS de capa 4 en el que el nodo malicioso establece los mismos datos de origen y destino de un segmento TCP. Los paquetes de ataque tienen indicadores S y utilizan el protocolo TCP. Un nodo atacado se bloqueará debido a que el mismo paquete es procesado repetidamente por la pila TCP.
W32. El ataque de gusano Blaster se propaga mediante el uso de la vulnerabilidad de saturación de búfer de la interfaz RPC DCOM de Microsoft Windows. Los ataques se dirigen solo a los puertos 135, 69 (TFTP) y 4444 (Kerberos).

El ataque Code Red Worm tiene como objetivo causar un problema de desbordamiento de búfer en un nodo de destino, de modo que comience a sobrescribir la memoria adyacente. Los paquetes se dirigen a la IP de origen y solo a 80 puertos (sin Secure Sockets Layer (SSL)); así es como se aplica el método HTTP GET.

El ataque de los bots de spam envía mensajes de spam o publica spam en plataformas de redes sociales o foros. Los paquetes se dirigen solo al puerto 25 (sin SSL). El ataque se caracteriza por la presencia de un número excesivamente grande de conexiones SMTP desde una dirección.

El ataque Reaper Worm comienza su última fase de escaneo una vez que la IP se pasa al proceso de explotación. El ataque Reaper está dirigido a los puertos TCP 81, 82, 83, 84, 88, 1080, 3000, 3749, 8001, 8060, 8080, 8081, 8090, 8443, 8880 y 10,000. Un ataque solo se registra cuando el paquete contiene el flujo TCP y no tiene protocolos UDP, ICMP o ICMP6.

El ataque de escaneo/propagación de puertos envía las solicitudes de los clientes a algunas direcciones de puerto del servidor, con el objetivo de descubrir un puerto activo y aprovechar un agujero de seguridad conocido. Un número anormal de conexiones de un host a uno o más hosts es el siguiente: varios puertos, una dirección; Un solo puerto, varias direcciones.

El ataque de fragmentación de paquetes es un tipo de ataque DoS, en el que el atacante sobrecarga una red aprovechando la fragmentación del datagrama.

3.1.1.3. Disponibilidad del conjunto de datos

Los datos de tráfico de red se capturan en los archivos de formato binario nfcapd. Los archivos nfcapd se recopilan en un solo archivo por semana durante dos períodos de captura. El tamaño medio de los archivos es de aproximadamente 1,35 GB (comprimidos). Los archivos nfcapd tienen todas las funciones de NetFlow, ampliadas con 19 funciones personalizadas de detección de ataques que comienzan desde el primer flujo del 06/03/2019 y el último flujo del 31/01/2020. Las direcciones IP de los nodos de la red se han anonimizado. La información de los remitentes (archivos de datos sin procesar de NetFlow) al recopilador se recibe en formato NetFlow v9 (rfc3954). Todos los valores se transfieren a la base de datos MySQL (base de datos NetFlow, servidor SQL).

Todos los archivos del conjunto de datos se pueden descargar gratuitamente desde el sitio web: https://dataset.litnet.lt.

La formación del conjunto de datos se resume en la Figura .

El preprocesador de datos selecciona 49 atributos que son específicos del protocolo NetFlow v9 (RFC 3954) para formar un conjunto de datos. El extensor de datos expande el conjunto de datos generado con campos de tiempo adicionales, marcas tcp, que luego se utilizan para identificar ataques. El conjunto de datos extendido se complementa con un conjunto de 15 atributos. El generador crea 19 atributos adicionales para el reconocimiento del tipo de ataque. Las combi-

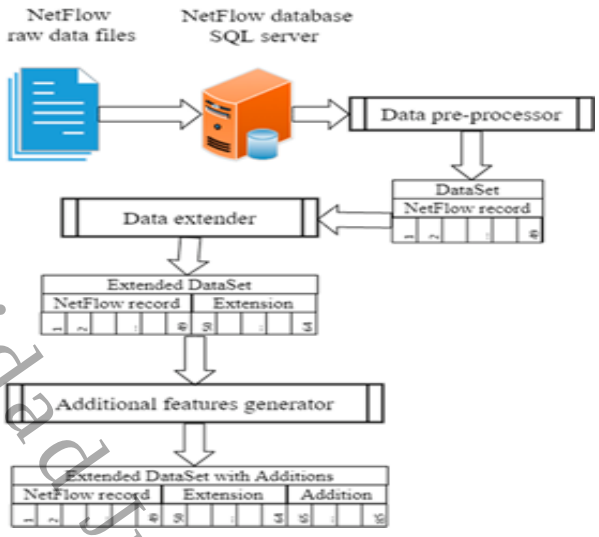


Figura 3.2. Formación del conjunto de datoset al, 2019

naciones de estos y los atributos de NetFlow se utilizan para detectar ataques.

Se agregaron dos campos adicionales para separar en el conjunto de datos, dónde se asigna el registro al ataque y qué tipo específico de ataque, y dónde está el tráfico de red normal. Por lo tanto, se tienen un total de 85 atributos.

3.1.1.4. Resumen

El conjunto de datos propuesto se recopiló en la red del mundo real, durante un 10 meses, y contiene ataques de red reales a través de la infraestructura de red de todo el país de Lituania con servidores en cuatro ubicaciones distribuidas geográficamente (ciudades). Como tal, el conjunto de datos de flujo de red propuesto tiene más ventajas que algunos de sus homólogos (como el conjunto de datos UNSW-NB 15 [59]), que generaron los ataques artificialmente y, por lo tanto, no contienen datos realistas.

3.2. Exploración y visualización de datos

El análisis exploratorio inicial del conjunto de datos de flujos de red LITNet tiene varios objetivos importantes. Estos objetivos se centran en entender a fondo la naturaleza de los datos y preparar el terreno para el entrenamiento del modelo que se pretende construir. Con este análi-

sis se logrará comprender la estructura del dato, las distribución del dato, identificar patrones y tendencias, detectar anomalías, comprensión del trafico y caracterización de flujos, seleccionar características relevantes o en su caso crear nuevas características que nos puedan proporcionar información adicional.

3.2.0.1. Estadísticas descriptivas de las clases cuantitativas

Tabla 3.1. Estadísticas descriptivas del conjunto de datos del tiempo

Estadísticas	**ID**	**ts_year**	**ts_month**	**ts_day**	**ts_hour**	**ts_minute**	**ts_second**
Conteo	39603674	39603674	39603674	39603674	39603674	39603674	39603674
Mean	2211817	2019.6807	4.0991	19.3377	10.2791	36.1187	28.8325
Std	1474588	0.4662	4.6625	12.1132	6.2965	12.3158	17.2860
Min	1	2019	1	1	0	0	0
25 %	948736.	2019	1	6	5	32	19
50 %	2094993	2020	1	25	12	36	37
75 %	3361644	2020	10	30	15	48	56
Max	6012524	2020	12	31	23	59	59

Tabla 3.2. Estadísticas descriptivas del conjunto de datos

Statistic	**te_year**	**te_month**	**te_day**	**te_hour**	**te_min**	**te_second**
Count	39603674	39603674	39603674	39603674	39603674	39603674
Mean	2019.6808	4.0989	19.3352	10.2727	36.2368	28.8977
Std	0.4662	4.6626	12.1135	6.2962	12.3088	17.2849
Min	2019	1	2	2	8	0
25 %	2019	1	3	3	31	19
50 %	2020	1	25	9	36	37
75 %	2020	10	30	15	47	55
Max	2020	12	31	21	56	59

Tabla 3.3. Estadísticas descriptivas de las variables seleccionadas

Statistic	td	sp	dp	fwd	stos	ipkt
Count	39603674	39603674	39603674	39603674	39603674	39603674
Mean	141.1806	27341.4701	21285.6138	0	36.2686	43.7545
Std	10748.4466	25572.1907	23243.3786	0	62.1982	1284.8202
Min	0	0	0	0	0	1
25 %	0	5060	1220	0	0	5
50 %	0	50876	31645.5	0	0	5
75 %	8.1300	56810	51614	0	128	15
Max	2347510.87	65535	65535	0	255	2516745

Tabla 3.4. Estadísticas descriptivas para tráfico de red y otros indicadores

Statistic	ibyt	opkt	obyt	_in	out	sas
Count	39603674	39603674	39603674	39603674	39603674	39603674
Mean	37370.3503	0	0	526.0924	459.2931	38989.7313
Std	1783880.2949	0	0	253.3576	283.2279	4777738.0567
Min	28	0	0	3	0	0
25 %	200	0	0	556	505	5650
50 %	440	0	0	649	558	14987
75 %	4850	0	0	707	707	63996
Max	3572836290	0	0	1030	1030	4294967295

Tabla 3.5. Estadísticas descriptivas para contadores y marcadores de red

Statistic	das	smk	dmk	dtos	_dir	svln
Count	39603674	39603674	39603674	39603674	39603674	39603674
Mean	472642.0824	0	0	0	0	0
Std	44537064.4958	0	0	0	0	0
Min	0	0	0	0	0	0
25 %	2847	0	0	0	0	0
50 %	2847	0	0	0	0	0
75 %	26496	0	0	0	0	0
Max	4294967295	0	0	0	0	0

Tabla 3.6. Estadísticas descriptivas para indicadores adicionales

Statistic	dvln	cl	sl	al	exid	attack_a
Count	39603674	39603674	39603674	39603674	39603674	39603674
Mean	0	0	0	0	1.3206	0.0803
Std	0	0	0	0	0.7493	0.2717
Min	0	0	0	0	1	0
25 %	0	0	0	0	1	0
50 %	0	0	0	0	1	0
75 %	0	0	0	0	2	1
Max	0	0	0	0	7	1

3.2.0.2. Resultados del análisis descriptivo

El análisis estadístico descriptivo permitió identificar que un total de doce (12) variables cuantitativas del conjunto de datos presentan una varianza nula, es decir, mantienen un valor constante igual a cero en todos los registros. Debido a que estas variables no contienen información útil ni aportan valor al proceso de aprendizaje del modelo, se ha decidido eliminarlas en esta etapa preliminar con el objetivo de optimizar el espacio de almacenamiento y reducir la complejidad computacional.

La Tabla 3.7 muestra el listado de las variables eliminadas por este motivo.

Tabla 3.7. Variables con varianza nula eliminadas del conjunto de datos

N.º	Nombre de la Variable
1	fwd
2	opkt
3	obyt
4	smk
5	dmk
6	dtos
7	_dir
8	svln
9	dvln
10	cl
11	sl
12	al

3.2.0.3. Estadísticas de la variables tipo objeto

Tabla 3.8. Resumen de IPs relevantes en la columna `sa`

IPs con mayor número de conteos		IPs con un único conteo	
IP	**Conteo**	**IP**	**Conteo**
104.152.52.31	1,084,835	5.107.46.87	1
104.152.52.18	933,108	5.75.29.30	1
193.219.88.36	857,629	47.39.57.214	1
80.82.77.132	837,140	45.17.45.121	1
80.82.77.221	721,418	58.96.56.237	1
193.219.81.138	603,638	58.72.58.90	1
80.82.64.73	369,315	59.101.242.136	1
104.152.52.21	281,841	5.13.2.54	1
141.98.11.12	274,123	54.243.217.33	1
193.219.162.82	232,196	5.80.61.161	1

Tabla 3.9. Conteos de valores para columnas de protocolo y banderas

Columna	Valor	Columna	Valor	Columna	Valor
`pr`	TCP: 29,152,801	`_flag1`	. : 39,416,607	`_flag2`	. : 22,762,180
	UDP: 9,445,158		(espacio): 187,050		A : 16,654,444
	ICMP: 740,871		U: 17		(espacio): 187,050
	IPv6: 166,281	`_flag3`	.: 32,199,958	`_flag4`	. : 38,191,373
	GRE: 33,968		P: 7,216,666		R: 1,225,251
	ESP: 32,208		0: 187,050		x: 187,050
	ICMP6: 23,688	`_flag5`	. : 24,507,770	`_flag6`	. : 37,307,002
	AH: 6,425		S : 14,908,854		F : 2,109,622
	OSPF: 1,152		c : 114,646		2 : 135,345
	IGMP: 480		d : 36,186		a : 32,814
	VRRP: 319		5 : 31,147		3 : 5,290
	PIM: 234		9 : 4,928		8 : 3,864
	IPIP: 89		4 : 85		b : 2,929
		7 : 1		e : 2,330	
		f : 1		4 : 2,075	
			6 : 1,933		
			9 : 244		
			c : 149		
			0 : 72		
			1 : 4		
			f : 1		

Tabla 3.10. Resumen de columnas con valor único en todo el conjunto de datos

Columna	Valor Único	Conteo
nh	0.0.0.0	39,603,674
nhb	0.0.0.0	39,603,674
ismc	00:00:00:00:00:00	39,603,674
odmc	00:00:00:00:00:00	39,603,674
idmc	00:00:00:00:00:00	39,603,674
osmc	00:00:00:00:00:00	39,603,674
mpls1	0-0-0	39,603,674
mpls2	0-0-0	39,603,674
mpls3	0-0-0	39,603,674
mpls4	0-0-0	39,603,674
mpls5	0-0-0	39,603,674
mpls6	0-0-0	39,603,674
mpls7	0-0-0	39,603,674
mpls8	0-0-0	39,603,674
mpls9	0-0-0	39,603,674
mpls10	0-0-0	39,603,674
ra	0.0.0.0	39,603,674
eng	0/0	39,603,674
tr	1970-01-01 03:00:00.000	39,603,674

Tabla 3.11. Resumen de ataques y flujos normales identificados en la red

Categoría	Normal	Blanco	Ataque/Anomalía
ICMP Destino IP	36,423,860	3,060,856	118,958 (icmp_smf)
ICMP Fuente IP	36,423,860	3,156,558	23,256 (icmp_f)
UDP Destino Puerto	36,423,860	3,086,231	93,583 (udp_f)
TCP Flag S	36,423,860	1,599,798	1,580,016 (tcp_syn_f)
TCP Flag NA	36,423,860	1,599,798	1,580,016 (tcp_syn_f)
TCP Flag NF	36,423,860	1,599,798	1,580,016 (tcp_syn_f)
TCP Flag NR	36,423,860	1,599,798	1,580,016 (tcp_syn_f)
TCP Flag NP	36,423,860	1,599,798	1,580,016 (tcp_syn_f)
TCP Flag NU	36,423,860	1,599,798	1,580,016 (tcp_syn_f)
TCP Destino Puerto	36,423,860	3,156,855	22,959 (http_f)
TCP SRC/DST Flag S	36,423,860	3,127,397	52,417 (tcp_land)
TCP SRC TFTP	36,423,860	3,155,523	24,291 (tcp_w32_w)
TCP SRC Kerberos	36,423,860	3,155,523	24,291 (tcp_w32_w)
TCP SRC RPC	36,423,860	3,155,523	24,291 (tcp_w32_w)
TCP DST SRC	36,423,860	1,924,112	1,255,702 (tcp_red_w)
SMTP Destino	36,423,860	3,179,067	747 (smtp_b)
UDP Puerto Rango	36,423,860	3,178,638	1,176 (udp_reaper_w)
Rango Destino Puerto	36,423,860	3,173,582	6,232 (tcp_udp_win_p)
UDP SRC Puerto 0	36,423,860	3,179,337	477 (udp_0)

Tabla 3.12. Conteos de tipos de ataques en la columna `attack_t`

Tipo de Ataque	Conteo
NORMAL	36,423,860
TCP SYN Flood	1,580,016
TCP Redirect	1,255,702
ICMP Source Quench	118,958
UDP Flood	93,583
TCP Land	52,417
TCP W32 Worm	24,291
ICMP Flood	23,256
HTTP Flood	22,959
TCP/UDP Win Probe	6,232
UDP Reaper	1,176
SMTP Bomb	747
UDP Zero	477

3.2.1. Visualización

A continuación se presentan gráficos de barras para representar la distribución de protocolos y otras características importantes sobre el tráfico.

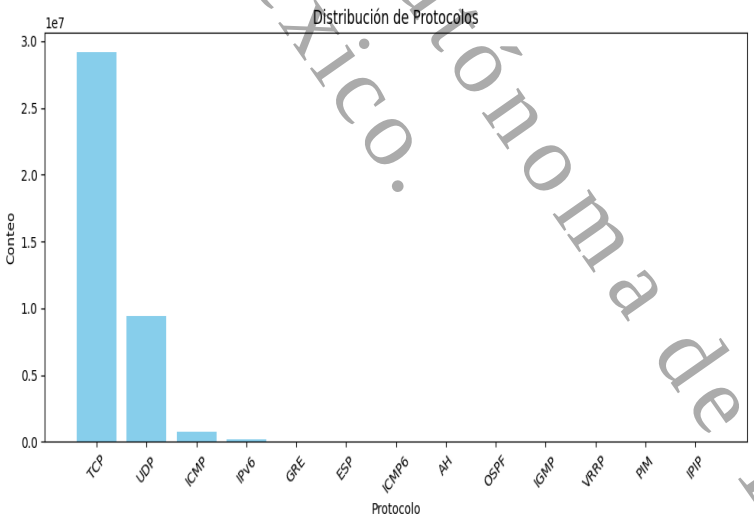


Figura 3.3. Histogramas del protocolo LITNETet al, 2019

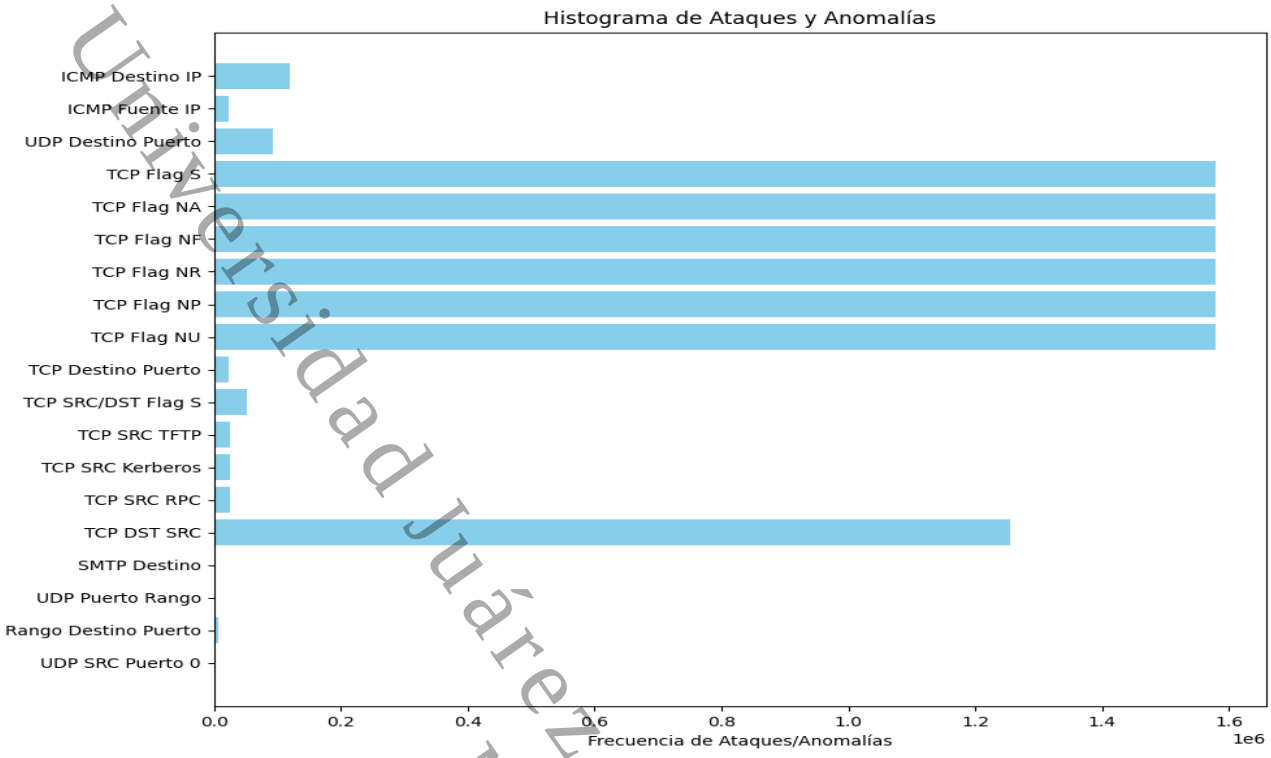


Figura 3.4. Histogramas del anomalías LITNETet al, 2019

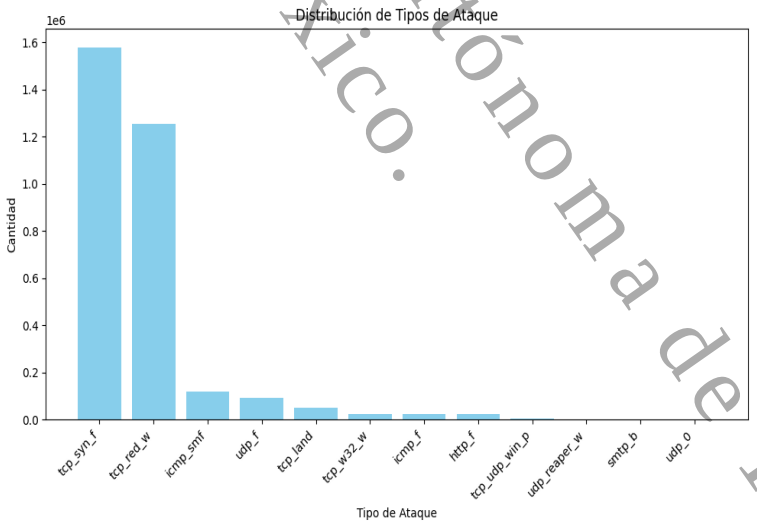


Figura 3.5. Tipos de ataques en LITNETet al, 2019

3.3. Transferencia de aprendizaje

3.3.1. Introducción

El modelo propuesto, denominado *CANET*, se basa en una arquitectura jerárquica que integra Redes Neuronales Convolucionales (*Convolutional Neural Networks, CNN*) con mecanismos de atención (*Attention Block*). Esta combinación permite la extracción eficaz de características espacio-temporales de los datos de tráfico de red, abordando desafíos como la alta tasa de falsos positivos y el desequilibrio de clases, aspectos comunes en los conjuntos de datos de detección de intrusiones.

En esta investigación, se aplica un enfoque de transferencia de aprendizaje específico, centrado en la **transferencia de arquitectura**. Dado que las características aprendidas en modelos preentrenados con datos sintéticos no son aplicables directamente a los datos realistas de Litnet, se optó por reutilizar la estructura del modelo CANET en lugar de los pesos preentrenados. Litnet presenta una mayor complejidad y diversidad de características en comparación con conjuntos de datos sintéticos como UNSW-NB15 o CICIDS 2017, lo cual hace que la transferencia de pesos sea ineficaz debido a las diferencias en la representación y el formato de los datos.

El formato de Litnet, basado en NetFlow 9, captura flujos completos de tráfico de red en tiempo real, incluyendo patrones temporales y relaciones jerárquicas que no se encuentran en los datos generados en entornos controlados. Este nivel de granularidad y secuencialidad demanda una adaptación del modelo desde cero, ya que los pesos preentrenados no capturan adecuadamente la complejidad y variabilidad inherente de los flujos de red reales.

Por lo tanto, el método de transferencia de arquitectura implica reutilizar la configuración de capas y los mecanismos de atención jerárquica desarrollados en CANET, mientras se entrena el modelo con los datos reales de Litnet. Este enfoque aprovecha el diseño modular y jerárquico del modelo original, adaptándolo a las características específicas del tráfico de red realista en Litnet. La transferencia de arquitectura permite una mejor detección y clasificación de ataques en este entorno complejo, mejorando la precisión y robustez del modelo para la detección de intrusiones.

En resumen, la transferencia de aprendizaje en este contexto se define como la reutilización de la arquitectura de CANET para ajustarse a las características espacio-temporales de los datos de Litnet. Al adaptar la estructura del modelo y entrenarlo desde cero, se logra una mayor eficacia

en la detección de ataques en un entorno de red más realista y dinámico, sin depender de pesos preentrenados que no reflejan adecuadamente las condiciones de Litnet.

3.3.2. Comparativa de conjuntos de datos para sistemas de detección de intrusos

La efectividad de un Sistema de Detección de Intrusos en la Red (NIDS) depende en gran medida de la calidad y características del conjunto de datos utilizado para su entrenamiento. A continuación, se analizan los conjuntos de datos más utilizados en la investigación de NIDS, destacando sus principales características y las diferencias con Litnet, que utiliza el formato NetFlow 9.

3.3.2.1. Características de los conjuntos de datos de referencia

Los siguientes conjuntos de datos son ampliamente utilizados en la investigación de NIDS, y cada uno presenta características particulares:

- **DDoS 2016:**
  - Simulado en un entorno controlado (NS2), incluye 4 tipos de ataques DoS/DDoS.
  - Ofrece 27 características y 5 clases de tráfico, con 734,627 registros.
- **UNSW-NB15:**
  - Generado por IXIA PerfectStorm, mezcla tráfico real y simulado.
  - Proporciona 49 características y 9 tipos de ataque, con 257,673 registros combinados de entrenamiento y prueba.
- **CICIDS 2017:**
  - Desarrollado por el Canadian Institute for Cybersecurity, contiene ataques multietapa como *Heartbleed*, DoS y DDoS.
  - Incluye 80 características y alrededor de 3 millones de registros.
- **UGR'16:**

- Recopilado por un ISP en España durante 4 meses, está diseñado para NIDS ciclostacionarios.
- Contiene 16,900 millones de flujos y 13 tipos de ataques, combinando tráfico real y sintético.

- **NSL-KDD:**
  - Derivado del KDD original, elimina elementos redundantes para evitar sesgos.
  - Ofrece 41 características y 4 tipos de ataque, con alrededor de 5 millones de registros.
- **CSE-CIC-IDS2018:**
  - Generado con perfiles sintéticos, incluye 6 tipos de ataques.
  - Contiene 84 características y alrededor de 20 millones de registros.

3.3.2.2. Comparativa con Litnet

El conjunto de datos Litnet, recopilado en una red nacional real, se diferencia significativamente de los conjuntos anteriores en varios aspectos clave:

- **Entorno Realista:** A diferencia de los conjuntos generados en entornos controlados (DDoS 2016, UNSW-NB15, CICIDS 2017), Litnet captura tráfico real de una red educativa y de investigación, lo que implica mayor complejidad y variabilidad en los datos.
- **Mayor Granularidad de Características:** Litnet incluye características de flujo más detalladas y temporales que conjuntos tradicionales como NSL-KDD y CSE-CIC-IDS2018, lo que requiere técnicas de segmentación y extracción más avanzadas para detectar relaciones jerárquicas en el tráfico de red.
- **Diversidad de Ataques:** Litnet registra tanto ataques conocidos como nuevas secuencias, representando la secuencia completa de los flujos de ataque, mientras que otros conjuntos suelen enfocarse en representaciones más estáticas.
- **Formato NetFlow 9:** Litnet utiliza NetFlow 9, un formato desarrollado por Cisco, que captura flujos de red completos con metadatos detallados, estadísticas de tráfico y características

a nivel de flujo. Esto ofrece una visión integral y continua del comportamiento de la red, adecuada para análisis en tiempo real, lo cual es esencial para detectar ataques avanzados.

3.3.2.3. Diferencias en el formato de datos

El uso de NetFlow 9 en Litnet resalta varias diferencias frente a los formatos de los otros conjuntos de datos:

- **NetFlow 9 en Litnet:**
  - **Formato Completo de Flujos:** Captura información agregada, como la cantidad de bytes y paquetes, tiempos de inicio y finalización, IPs de origen y destino, y protocolos.
  - **Granularidad Temporal y Secuencial:** Proporciona información detallada sobre la secuencia temporal del tráfico, lo que facilita la identificación de patrones de red en tiempo real.
  - **Aplicabilidad en Redes Grandes y Reales:** Permite analizar redes grandes y complejas, lo cual es crucial para una detección precisa de intrusos en entornos más realistas.
- **Formatos en Otros Conjuntos de Datos:**
  - **DDoS 2016 y UNSW-NB15:** Utilizan formatos simplificados centrados en atributos de paquetes o segmentos cortos de conexión, lo cual limita su capacidad de generalización en entornos más dinámicos.
  - **CICIDS 2017 y CSE-CIC-IDS2018:** Emplean el formato de CICFlowMeter, que captura características de flujos pero con menos granularidad y detalle temporal que NetFlow 9.
  - **UGR'16:** Contiene flujos de red similares a NetFlow pero con un enfoque más agregado, lo que puede limitar la detección de intrusiones avanzadas.
  - **NSL-KDD:** Se enfoca en características estadísticas y de contenido derivadas de la inspección de paquetes, ofreciendo menos profundidad temporal.

3.3.2.4. Diferencias clave en el uso de NetFlow 9

Las diferencias clave que presenta Litnet gracias al uso de NetFlow 9 incluyen:

- **Granularidad Superior:** Permite capturar tráfico en mayor detalle, lo cual es fundamental para la detección de ataques complejos y distribuidos en el tiempo.
- **Datos en Tiempo Real:** La estructura de Litnet facilita la captura de datos en tiempo real, crucial para la detección rápida de intrusiones.
- **Mayor Flexibilidad:** NetFlow 9 ofrece una mejor adaptabilidad para capturar diversos tipos de tráfico y comportamientos, lo que lo hace ideal para entornos de red complejos y variables.

3.3.2.5. Resumen comparativo de los conjuntos de datos

La Tabla 3.13 resume las características clave de los conjuntos de datos analizados, incluyendo el tipo de formato de captura de red.

Tabla 3.13. Resumen de Conjuntos de Datos de Detección de Intrusos

Conjunto de Datos	Características	Clases de Ataque	Registros	Duración	Formato de Captura
DDoS 2016	27	4	734,627	31 horas	Simplificado (Paquetes)
UNSW-NB15	49	9	257,673	31 horas	Mixto (Paquetes y Conexión)
CICIDS 2017	80	15	∼3 millones	5 días	CICFlowMeter (Flujos)
UGR'16	-	13	16,900 millones	4 meses	Flujos Agregados
NSL-KDD	41	4	∼5 millones	-	Conexiones (Estadísticas)
CSE-CIC-IDS2018	84	6	∼20 millones	-	CICFlowMeter (Flujos)
Litnet	33	13	792,073	Real (Variable)	NetFlow 9 (Flujos Completo)

3.3.2.6. Conclusión

El uso de NetFlow 9 en Litnet requiere un tratamiento especializado para aprovechar sus datos más ricos y en tiempo real, lo que incluye técnicas avanzadas de codificación, segmentación y normalización. Estas diferencias destacan la necesidad de una arquitectura de modelo adaptada y optimizada para la detección de intrusos en entornos de red más realistas, garantizando una mayor precisión y fiabilidad.

3.3.3. Arquitectura original del modelo

El modelo inicial desarrollado para la detección de intrusiones se implementó utilizando **Keras y TensorFlow**, frameworks ampliamente utilizados para el desarrollo de redes neuronales. La arquitectura original se basa en una red neuronal convolucional (*Convolutional Neural Network, CNN*) con mecanismos de atención, diseñada para procesar el conjunto de datos sintéticos. Estos conjuntos de datos se generan en un entorno controlado, lo que significa que las características de tráfico de red se capturaron en un entorno estático, sin la variabilidad ni la complejidad que presentan los datos reales de red. Esta estructura estática simplifica la extracción de patrones, pero limita la capacidad del modelo para generalizarse a flujos de red más complejos y dinámicos.

La combinación de capas convolucionales y bloques de atención permite la extracción eficaz de características espacio-temporales, lo cual es esencial para mejorar la precisión en la clasificación de diferentes tipos de tráfico de red, especialmente en un entorno con desequilibrio de clases.

A continuación, se detalla la estructura y componentes del modelo original:

3.3.3.1. Preprocesamiento de los datos

El preprocesamiento de los datos fue un paso crítico para garantizar la compatibilidad con la red convolucional, e incluyó las siguientes etapas:

- **Codificación One-Hot:** Las columnas categóricas del conjunto de datos, como *proto*, *state* y *service*, se convirtieron en representaciones numéricas mediante codificación one-hot, permitiendo al modelo procesar estas características de manera efectiva.
- **Normalización:** Las características numéricas se normalizaron en un rango de [0, 1] para estabilizar el proceso de entrenamiento y mejorar la convergencia del modelo.
- **Separación de Características y Etiquetas:** Después de la codificación y normalización, se separaron las características de la variable objetivo (*attack_cat*), preparando los datos para el entrenamiento y la validación.

3.3.3.2. Estructura del modelo

El diseño de la red CNN con atención se centró en la extracción progresiva de características a través de capas convolucionales, integradas con bloques de atención para mejorar la identificación de patrones de tráfico relevantes:

- **Entrada:** El modelo acepta como entrada un tensor tridimensional de tamaño $(batch\_size, 196, 1)$, donde 196 representa la longitud de la secuencia de características y 1 indica el canal de entrada.
- **Capas Convolucionales y Max-Pooling:**
  - **Primera Capa Convolucional:** Aplica 64 filtros con un tamaño de núcleo de 64, seguido de una capa de max-pooling con un tamaño de núcleo de 4 y normalización por lotes.
  - **Segunda Capa Convolucional:** Contiene 128 filtros con un tamaño de núcleo de 64, seguido de max-pooling de tamaño 2 y normalización por lotes.
  - **Tercera Capa Convolucional:** Utiliza 256 filtros con un tamaño de núcleo de 64, seguido de max-pooling de tamaño 2 y normalización por lotes.
- **Mecanismo de Atención:** Después de cada capa de normalización, se implementa un bloque de atención. La atención se aplica en tres niveles, permitiendo al modelo asignar pesos diferentes a las características más relevantes en cada capa de abstracción. Esto se logra mediante la función de atención de Keras, que pondera las características de entrada en función de su relevancia para la tarea de clasificación.
- **Capa de Salida:** La salida de la red se aplana y se conecta a una capa densa con 10 neuronas, una para cada clase del conjunto de datos UNSW-NB15, utilizando una activación *softmax* para la clasificación.

3.3.3.3. Función de pérdida EQLv2

Para abordar el desequilibrio de clases presente en UNSW-NB15, se utilizó la función de pérdida *Equalization Loss v2 (EQLv2)*. Esta función ajusta dinámicamente los pesos de las clases

durante el entrenamiento, enfocándose en mejorar la detección de clases minoritarias. EQLv2 considera gradientes positivos y negativos para ajustar los pesos según la representación de cada clase en el conjunto de datos. La implementación se realizó mediante una clase personalizada en TensorFlow, integrada directamente en el proceso de entrenamiento.

3.3.3.4. Proceso de entrenamiento y validación

El modelo original fue entrenado utilizando validación cruzada estratificada con 10 pliegues, asegurando una evaluación robusta del rendimiento. El proceso de entrenamiento involucró:

- **División de Datos:** El conjunto de datos se dividió en 10 pliegues, utilizando uno para validación y los restantes para entrenamiento en cada iteración.
- **Optimización:** Se utilizó el optimizador Adam con una tasa de aprendizaje predeterminada, entrenando el modelo durante 150 épocas en cada pliegue con un tamaño de batch de 256.
- **Evaluación del Rendimiento:** Después de cada pliegue, se evaluó el rendimiento utilizando métricas como precisión, *recall*, tasa de falsos positivos (FPR) y F1-Score. También se generó un informe de clasificación y una matriz de confusión para evaluar la capacidad del modelo de identificar correctamente las diferentes categorías de intrusiones.

3.3.3.5. Resultados del modelo original

El modelo original demostró un rendimiento prometedor en la identificación de múltiples categorías de intrusiones dentro del conjunto de datos UNSW-NB15. Sin embargo, se observó que su capacidad de generalización se vio limitada, especialmente en la detección de clases minoritarias, debido a la naturaleza del conjunto de datos, que se generó en un entorno controlado y estático. Esto resalta la necesidad de una adaptación más compleja para trabajar con datos de red más realistas, como los de Litnet.

3.3.4. Transferencia de la arquitectura

3.3.4.1. Transferencia de arquitectura para datos reales de flujo de red

La adaptación de la arquitectura del modelo inicial a los datos de red reales se realizó mediante una estrategia de **transferencia de arquitectura**, la cual permite reutilizar la estructura del modelo original, pero ajustándola a las características específicas del conjunto de datos Litnet. El modelo propuesto, denominado *CANET*, se basa en una arquitectura jerárquica que integra Redes Neuronales Convolucionales (*Convolutional Neural Networks, CNN*) con mecanismos de atención (*Attention Block*). Esta combinación facilita la extracción eficaz de características espacio-temporales de los datos de tráfico de red, abordando los desafíos relacionados con la alta tasa de falsos positivos y el desequilibrio de clases en los conjuntos de datos de detección de intrusiones.

Estructura jerárquica El diseño de CANET sigue una estructura jerárquica, donde múltiples **Bloques de CNN con Atención (CA)** se apilan para capturar características complejas en diferentes niveles de abstracción. Esta arquitectura modular permite la adaptación del modelo a distintos tipos de tráfico de red y patrones de ataque, asegurando que tanto las características de bajo nivel como las de alto nivel sean capturadas y procesadas de manera adecuada.

Preprocesamiento Antes del proceso de extracción de características, se realizó un preprocesamiento exhaustivo de los datos, que incluyó:

- **Categorización:** Las etiquetas de texto presentes en los datos, como los protocolos específicos (*http*, *smtp*), se convirtieron en representaciones numéricas, lo que permite su procesamiento adecuado por la red neuronal.
- **Normalización Min-Max:** Los valores de las características de entrada se escalaron a un rango de [0, 1] mediante normalización Min-Max, asegurando que todas las características contribuyan de manera equilibrada al proceso de aprendizaje, mejorando la estabilidad y velocidad de la propagación hacia atrás.

Bloques de CNN con atención (CA) El núcleo de CANET es el **Bloque de CNN con Atención (CA)**, una unidad modular que combina capas de CNN con mecanismos de atención en cada nivel de la red:

- **Capas Convolucionales (CNN):** Las capas CNN se encargan de extraer características espaciales del tráfico de red. Aplican filtros convolucionales para capturar patrones locales y estructuras en los datos, esenciales para identificar firmas de ataques específicos.
- **Mecanismo de Atención:** Después de cada capa convolucional, se incorpora un mecanismo de atención, que enfoca el aprendizaje en características temporales críticas, asegurando que las características relevantes no se pierdan durante el proceso de extracción espacial. La atención permite que el modelo asigne mayor importancia a las características cruciales para la clasificación, mientras reduce la influencia de características menos relevantes.
- **Integración del Bloque CA:** La integración de CNN y atención dentro de cada Bloque CA permite que el modelo aprenda de manera efectiva múltiples niveles de características espacio-temporales, lo que es vital para la detección precisa de intrusiones en redes de gran escala.

Función de pérdida: Equalization Loss v2 (EQLv2) Para abordar el desequilibrio de clases presente en los datos de Litnet, se implementó la función de pérdida **EQLv2**. Este método ajusta dinámicamente los pesos de las clases minoritarias durante el entrenamiento, aumentando la atención del modelo hacia las clases menos representadas. EQLv2 recalibra los gradientes positivos y negativos de manera independiente para cada clase, equilibrando así la influencia de las muestras de clases mayoritarias y minoritarias. Esta estrategia elimina la necesidad de técnicas de sobremuestreo o submuestreo, reduciendo el costo computacional y mejorando la estabilidad del entrenamiento.

3.3.5. Fórmulas de EQLv2

La pérdida EQLv2 se define como:

$$L_{EQLv2} = \frac{\lambda}{N} \sum_{i=1}^{N} \text{CrossEntropy}(\hat{y}_i, y_i; w_i^{\text{pos}}, w_i^{\text{neg}}), \tag{3.1}$$

donde:

- N es el tamaño del lote.
- λ es un factor de ponderación.
- $\hat{y}_i$ son las predicciones del modelo para el ejemplo i.
- y_i es la etiqueta verdadera para el ejemplo i.
- w_i^{pos} y w_i^{neg} son los pesos dinámicos para gradientes positivos y negativos, definidos como:

$$w_c^{\text{pos}} = 1 + \alpha \cdot (1 - \text{map}(x_c)) \cdot f_c, \tag{3.2}$$

$$w_c^{\text{neg}} = \text{map}(x_c), \tag{3.3}$$

donde:

- c es el índice de la clase.
- $\text{map}(x_c) = \frac{1}{1+e^{-\gamma(x_c-\mu)}}$ es una función logística que ajusta la contribución dinámica de los gradientes acumulados.
- $x_c = \frac{\text{PosGrad}_c}{\text{NegGrad}_c+\epsilon}$, la relación entre los gradientes positivos y negativos acumulados para la clase c.
- $f_c = \frac{\sum_{j=1}^{N} f_j}{f_c+\epsilon}$, una ponderación inversa a la frecuencia de la clase c en el lote.

La pérdida de entropía cruzada ponderada se calcula como:

$$\text{CrossEntropy}(\hat{y}_i, y_i; w^{\text{pos}}, w^{\text{neg}}) = -\sum_{c=1}^{C} \left[y_{ic} \cdot w_c^{\text{pos}} \cdot \log(\sigma(\hat{y}_{ic})) + (1 - y_{ic}) \cdot w_c^{\text{neg}} \cdot \log(1 - \sigma(\hat{y}_{ic})) \right], \tag{3.4}$$

donde:

- C es el número total de clases.
- y_{ic} indica si la clase c es la verdadera para el ejemplo i.
- σ es la función sigmoide.

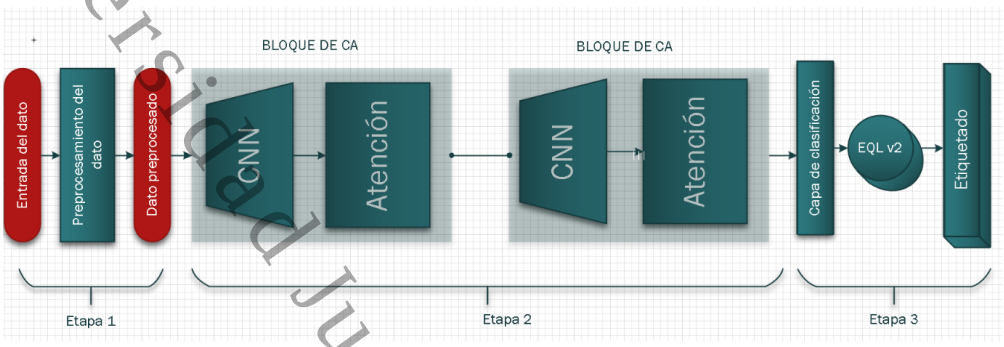


Figura 3.6. Arquitectura del modelo propuesto: CANET

Diferencias en la arquitectura entre el modelo original y el modelo ajustado La adaptación del modelo original a Litnet implicó cambios significativos en la estructura de las capas convolucionales, el mecanismo de atención y la implementación de la función de pérdida:

- **Capas convolucionales:**
  - **Modelo original:** El modelo original utilizaba tres capas convolucionales seguidas de operaciones de *max-pooling*, con un mecanismo de atención en cada nivel de características.
  - **Modelo Ajustado:** En CANET, se mantuvo la estructura base de las capas convolucionales, pero se ajustaron las dimensiones de las capas de atención jerárquica para adaptarse a las características específicas de Litnet.
- **Atención jerárquica:**
  - **Modelo original:** La atención se implementó usando la función de atención de Keras, optimizada para datos estáticos.
  - **Modelo ajustado:** En CANET, se implementó atención jerárquica directamente en PyTorch, permitiendo una asignación más flexible y precisa de pesos de atención en función de las características temporales y de flujo de Litnet.

Algorithm 1 Entrenamiento con pérdida EQLv2

Entrada: Conjunto de datos D con muestras (x_i, y_i), modelo f con parámetros θ, número de clases C, hiperparámetros γ, μ, α, *loss_weight*.
Salida: Parámetros optimizados del modelo θ.
Inicializar gradientes positivos $\_pos\_grad \leftarrow 0$ (vector de tamaño C);
Inicializar gradientes negativos $\_neg\_grad \leftarrow 0$ (vector de tamaño C);
Inicializar el contador de frecuencia por clase $class\_counts \leftarrow 0$ (vector de tamaño C);
for cada época **do**
    Reiniciar $\_pos\_grad$ y $\_neg\_grad$ a 0;
    for cada lote $B = \{(x_i, y_i)\}$ **do**
        Propagación hacia adelante:
        Predecir logits $pred\_logits \leftarrow f(x_i; \theta)$;
        Calcular probabilidades por clase $prob \leftarrow \text{sigmoid}(pred\_logits)$;
        Calcular pesos dinámicos:
        Actualizar frecuencias por clase $class\_counts[y] \leftarrow class\_counts[y] + 1$;
        Calcular pesos de frecuencia por clase:
        $class\_weights \leftarrow \frac{\sum class\_counts}{class\_counts + \epsilon}$;
        Normalizar $class\_weights$;
        Calcular pesos dinámicos:
        $neg\_w \leftarrow map\_func\left(\frac{\_pos\_grad}{\_neg\_grad + \epsilon}\right)$;
        $pos\_w \leftarrow 1 + \alpha \cdot (1 - neg\_w) \cdot class\_weights$;
        Calcular pérdida:
        Pérdida de entropía cruzada con pesos dinámicos:
        $cls\_loss \leftarrow \text{CrossEntropy}(pred\_logits, y, \text{pesos} = (pos\_w + neg\_w))$;
        Normalizar pérdida:
        $cls\_loss \leftarrow \frac{\sum cls\_loss}{\text{tamaño\_lote}}$;
        Propagación hacia atrás:
        Calcular gradientes $grad \leftarrow \nabla(cls\_loss)$;
        Actualizar parámetros del modelo:
        $\theta \leftarrow \theta - \eta \cdot grad$;
        Actualizar acumuladores de gradientes:
        $target\_one\_hot \leftarrow \text{one\_hot\_encoding}(y, C)$;
        $\_pos\_grad \leftarrow \_pos\_grad + grad \cdot target\_one\_hot \cdot pos\_w$;
        $\_neg\_grad \leftarrow \_neg\_grad + grad \cdot (1 - target\_one\_hot) \cdot neg\_w$;
    end for
    Normalizar acumuladores de gradientes:
    $pos\_neg\_ratio \leftarrow \frac{\_pos\_grad}{\_neg\_grad + \epsilon}$;
end for

- **Función de pérdida:**
  - **EQLv2 en ambos modelos:** Se utilizó la función de pérdida *Equalization Loss v2* para ajustar dinámicamente los pesos de las clases.
  - **Modelo Ajustado:** La implementación de EQLv2 en PyTorch permitió una mayor precisión en la acumulación de gradientes, lo que mejoró la detección de clases minoritarias en Litnet.

Comparación de la preparación y forma de los datos de entrada El proceso de preparación de los datos y la estructura del tensor de entrada difería entre la implementación original en TensorFlow y la adaptación en PyTorch:

- **TensorFlow:**
  - Los datos de UNSW-NB15 se prepararon para una red convolucional 1D, con codificación *one-hot* y normalización previa, generando una forma de entrada de $(batch\_size, num\_features, 1)$
- **PyTorch:**
  - En la adaptación para Litnet, se utilizó *CatBoostEncoder* para la codificación de columnas categóricas, aplicando la normalización después de la codificación y ajustando la entrada a $(batch\_size, 1, num\_features)$ para compatibilidad con *Conv1d*.

Tabla 3.14. Comparación de la forma de los datos de entrada en TensorFlow y PyTorch

Aspecto	**TensorFlow**	**PyTorch**
Estructura de Entrada	$(batch\_size, num\_features, 1)$	$(batch\_size, 1, num\_features)$
Codificación	*One-hot*	*CatBoostEncoder*
Normalización	Previa a la reorganización	Después de la codificación
Ajuste del Canal	Canal como última dimensión	Canal como primera dimensión
Compatibilidad	Keras *Conv1D*	PyTorch *Conv1d*

La principal diferencia radica en la ordenación de las dimensiones del tensor de entrada. TensorFlow coloca el canal al final de la secuencia, mientras que PyTorch lo coloca primero, lo que requirió ajustes específicos para asegurar la compatibilidad con la arquitectura convolucional en cada marco de trabajo.

Conclusión de la adaptación de la arquitectura La transferencia de arquitectura permitió preservar elementos clave del modelo original, adaptándolos a las características temporales y de flujo de Litnet. Al entrenar el modelo desde cero con datos de red reales, se logró una mejor generalización en la detección de intrusiones en entornos más dinámicos y realistas.

3.3.6. Comparación de implementación: TensorFlow vs. PyTorch

La transición del modelo de TensorFlow a PyTorch para el entrenamiento desde cero aportó beneficios significativos en términos de flexibilidad, eficiencia y capacidad de depuración. Esta comparación es relevante para entender cómo la elección del framework influye en el desarrollo de modelos de detección de intrusos. A continuación, se describen las principales diferencias entre ambas implementaciones:

- **Paradigma de Programación:**
  - **TensorFlow (Modelo Original):**
    - La implementación inicial se realizó en TensorFlow 2.x, que utiliza un enfoque de programación declarativa a través de la API de Keras, facilitando la construcción rápida de prototipos.
    - Sin embargo, el enfoque de ejecución diferida (*eager execution*) puede hacer que la depuración sea menos intuitiva, dificultando el monitoreo del flujo de datos y la interpretación de errores en tiempo real.
  - **PyTorch (Modelo Ajustado):**
    - PyTorch adopta un enfoque de ejecución dinámica, lo cual permite una depuración más sencilla y eficiente durante cada paso del entrenamiento.
    - La implementación de la atención jerárquica y la función de pérdida *EQLv2* en PyTorch resultó más flexible, ya que el framework permite realizar modificaciones más detalladas y controladas a nivel de capas y operaciones matemáticas.
- **Manejo de la pérdida y gradientes:**
  - **TensorFlow:**

- La acumulación de gradientes y el manejo de la pérdida *EQLv2* en TensorFlow requirieron una implementación más abstracta, limitando la capacidad de monitorear cambios graduales en la ponderación de las clases durante el entrenamiento.

- **PyTorch:**
  - PyTorch permitió una implementación más detallada de la pérdida *EQLv2*, con un monitoreo más claro de los gradientes positivos y negativos, lo cual resultó esencial para comprender el comportamiento del modelo en situaciones de desbalance de clases.
  - La capacidad de PyTorch para monitorear y actualizar los gradientes de manera directa facilitó la optimización dinámica de la pérdida, mejorando el ajuste del modelo a las clases minoritarias.

En resumen, la transición a PyTorch no solo mejoró la adaptabilidad del modelo al permitir un control más preciso del flujo de datos y de los gradientes durante el entrenamiento, sino que también facilitó una mayor visualización de los resultados internos del modelo, lo cual resultó crítico para ajustar de manera más efectiva el balance de clases y la detección de intrusiones. Además, el uso de PyTorch permite un crecimiento más acelerado en la comunidad científica debido a su popularidad y su mayor compatibilidad con librerías de análisis de gradientes y visualización, lo que facilita la reproducibilidad y la colaboración en futuros trabajos de investigación.

3.3.7. Síntesis de la sección de transferencia de aprendizaje

El enfoque de **transferencia de arquitectura** demostró ser una estrategia efectiva para adaptar el modelo original a las características únicas de los datos reales de flujo de red en Litnet. A diferencia de la transferencia de pesos preentrenados, que no fue viable debido a la naturaleza estática de los conjuntos de datos sintéticos utilizados inicialmente, la transferencia de arquitectura permitió reutilizar la estructura jerárquica del modelo *CANET*. Esto facilitó la captura de características espacio-temporales más complejas y diversas presentes en Litnet, mejorando la detección de intrusiones en un entorno de red más realista y dinámico.

La implementación de esta transferencia de arquitectura desde cero, junto con ajustes específicos en las capas de atención y en la función de pérdida *EQLv2*, no solo incrementó la

adaptabilidad y precisión del modelo, sino que también permitió manejar de manera más eficiente el desbalance de clases y la variabilidad de los flujos de red. Además, el uso de PyTorch como framework potenció el proceso de adaptación, gracias a su capacidad para gestionar de manera más explícita el flujo de datos y los gradientes, lo que resultó en una mayor visibilidad de los resultados internos del modelo.

En conclusión, la transferencia de arquitectura no solo optimizó la eficacia del modelo para detectar intrusiones en entornos de red reales, sino que también validó la importancia de ajustar la estructura de un modelo de red neuronal a los datos específicos del dominio. Este enfoque abre nuevas oportunidades para futuras investigaciones, al establecer una metodología clara para adaptar modelos preexistentes a contextos más complejos y dinámicos, maximizando su aplicabilidad y rendimiento en situaciones del mundo real.

Preparación de los datos para el modelo

El preprocesamiento de los datos del conjunto Litnet es un paso esencial para adecuar los datos de entrada a la red convolucional con atención jerárquica. Dada la complejidad y granularidad de Litnet, se requieren múltiples etapas para transformar los datos brutos en un formato compatible con el modelo. A continuación, se detalla cada paso clave del preprocesamiento.

Algorithm 2 Pipeline de Preprocesamiento

Require: Conjunto de datos D con características y variable objetivo sin procesar.
Ensure: Tensores normalizados para los conjuntos de entrenamiento, validación y prueba.

1: **Cargar** el conjunto de datos D
2: Extraer columnas categóricas C
3: Crear columnas de timestamps $timestamp$ y $timestamp2$ combinando año, mes, día, hora, minuto y segundo
4: Convertir $timestamp$ y $timestamp2$ a formato datetime
5: Establecer $timestamp$ como índice
6: Eliminar columnas irrelevantes de D
7: Codificar la variable objetivo $attack\_t$ como $attack\_t\_cat$ usando LabelEncoder
8: Aplicar Codificación CatBoost a las columnas categóricas C
9: Normalizar las características utilizando escalado min-max:
$X' = \frac{X - \text{mín}(X)}{\text{máx}(X) - \text{mín}(X)}$
10: Dividir los datos en conjuntos de entrenamiento, validación y prueba (70 %, 15 %, 15 %) con estratificación
11: Convertir X_{train}, X_{val}, y X_{test} a tensores de PyTorch con forma $(n\_samples, 1, n\_features)$
12: Guardar todos los tensores en disco para uso futuro

$$\hat{y}_c = \frac{\sum_{i \in D_c} y_i}{|D_c|} \tag{3.5}$$

Donde:

- $\hat{y}_c$: Valor codificado para la categoría c.
- D_c: Conjunto de datos donde la característica categórica toma el valor c.
- y_i: Valor de la variable objetivo asociado a la muestra i dentro de D_c.
- $|D_c|$: Número total de muestras en D_c.

Descripción: La codificación CatBoost reemplaza cada categoría c con la media de los valores de la variable objetivo y para todas las muestras en D_c. Esto preserva la relación estadística entre la categoría y la variable objetivo, especialmente útil para manejar características categóricas con alta cardinalidad. Se pueden aplicar técnicas como suavizado o validación cruzada para evitar el sobreajuste.

$$X' = \frac{X - \text{mín}(X)}{\text{máx}(X) - \text{mín}(X)} \tag{3.6}$$

Donde:

- X': Valor normalizado de la característica.
- X: Valor original de la característica.
- $\text{mín}(X)$: Valor mínimo de la característica en el conjunto de datos.
- $\text{máx}(X)$: Valor máximo de la característica en el conjunto de datos.

Descripción: La normalización min-max escala los valores de una característica para que estén dentro del rango $[0, 1]$. Esto asegura que todas las características tengan escalas comparables, evitando que las características con rangos más grandes dominen a otras. También mejora la estabilidad numérica durante la optimización, particularmente en algoritmos basados en gradientes como las redes neuronales.

3.3.7.1. Carga y limpieza de datos

La primera fase del preprocesamiento involucra la carga y limpieza inicial de los datos. Esto se realiza para eliminar características innecesarias y reducir la dimensionalidad, mejorando la eficiencia del modelo:

- **Carga de datos:**
  - Se utiliza la librería `pandas` para cargar el archivo CSV de Litnet, que contiene flujos de red etiquetados con múltiples características.
  - El conjunto de datos incluye tanto características numéricas como categóricas, así como información temporal detallada para cada flujo.
- **Eliminación de columnas irrelevantes:**
  - Se eliminan columnas que no proporcionan valor agregado al proceso de clasificación, tales como `ID`, `ts_year`, `ts_month`, `ts_day`, entre otras, las cuales solo representan partes individuales de la marca de tiempo y no son necesarias después de la conversión a `timestamp`.
  - También se descartan características relacionadas con campos específicos de protocolos (`icmp_src_ip`, `tcp_f_s`, `smtp_dst`, etc.), ya que estas características son altamente específicas y presentan un valor informativo limitado para la detección de múltiples tipos de ataques.
  - La selección de características se basa en la reducción del ruido en el modelo y la mejora de la velocidad de procesamiento sin comprometer la precisión de la clasificación.

3.3.7.2. Conversión y gestión de timestamps

Dado que Litnet tiene una estructura de datos altamente temporal, la gestión de las marcas de tiempo es fundamental:

- **Conversión a timestamps:**

- Se combinan las columnas individuales de año, mes, día, hora, minuto y segundo en una sola columna denominada `timestamp`, que representa la marca de tiempo inicial de cada flujo.
- La conversión a formato `datetime` permite una manipulación más precisa y facilita el reordenamiento de los datos en el dominio del tiempo, lo cual es crucial para el análisis secuencial en redes neuronales.
- Se crea una segunda columna `timestamp2` que representa el final del flujo, proporcionando una referencia adicional para la segmentación temporal de los datos.

- **Reordenamiento temporal:**
  - El conjunto de datos se reordena utilizando la columna `timestamp` como índice, asegurando que los flujos se procesen en un orden cronológico correcto, lo que es esencial para capturar la secuencia temporal de los eventos de red.
  - Esta reorganización permite que la red convolucional con atención jerárquica identifique patrones temporales y dependencias a lo largo del tiempo, mejorando la precisión de la clasificación.

3.3.7.3. Codificación de la variable objetivo

Para que la red pueda clasificar los tipos de ataque correctamente, se realiza una codificación adecuada de la variable objetivo:

- **Label Encoding:**
  - La variable objetivo original, denominada `attack_t`, se codifica mediante *Label Encoding*, lo cual transforma las etiquetas de texto en valores numéricos (por ejemplo, `Normal`=0, `DoS`=1, `Smurf`=2, etc.).
  - La nueva columna se denomina `attack_t_cat` y se utiliza como variable objetivo para la clasificación durante el entrenamiento del modelo.
  - La codificación de la variable objetivo asegura que la red neuronal interprete correctamente las diferentes clases de ataque.

3.3.7.4. Codificación de características categóricas

Litnet contiene múltiples características categóricas que deben ser adecuadamente codificadas para su procesamiento por parte de la red convolucional:

- **Target Encoding:**
  - Se aplica el *Target Encoding* a las columnas categóricas detectadas en el conjunto de datos, lo cual permite convertir valores categóricos en valores numéricos continuos, manteniendo la información relevante.
  - Este método de codificación asigna un valor promedio de la variable objetivo a cada categoría, lo que es particularmente útil en conjuntos de datos desbalanceados como Litnet, ya que ayuda a capturar la correlación entre características categóricas y la clase objetivo.
  - La codificación se realiza usando la librería `CatBoostEncoder` para mejorar la precisión del modelo, ya que esta técnica reduce la varianza introducida por las características categóricas de alta cardinalidad.

3.3.7.5. Normalización de las características

La normalización de las características es crucial para el entrenamiento eficaz de la red:

- **Normalización Min-Max:**
  - Se aplica la normalización min-max a todas las características numéricas, lo que asegura que todas las entradas estén escaladas dentro de un rango de 0 a 1, lo cual facilita la convergencia de la red durante el entrenamiento.
  - La normalización mejora la estabilidad numérica y la eficiencia computacional del modelo, asegurando que ninguna característica domine a otra debido a diferencias de escala.
  - Esta técnica también es fundamental para mejorar la capacidad del modelo de manejar la variabilidad en las características de red.

3.3.7.6. Preparación para pyTorch

Finalmente, los datos se preparan para su entrada en la red convolucional utilizando PyTorch:

- **Conversión a Tensores:**
  - Las características (`X`) se convierten a un arreglo de NumPy y se reorganizan en el formato $(\text{número de muestras}, 1, \text{número de características})$, lo cual es requerido por PyTorch para redes convolucionales 1D.
  - La variable objetivo (`y`) se convierte a un tensor de tipo `long` (enteros largos), ya que es necesario para tareas de clasificación en PyTorch.
  - Esta conversión garantiza que los datos sean compatibles con las operaciones tensoriales de PyTorch, facilitando el entrenamiento eficiente de la red convolucional con atención jerárquica.

3.3.7.7. Justificación del preprocesamiento de Litnet

El preprocesamiento de Litnet es más complejo en comparación con otros conjuntos de datos de detección de intrusos debido a su estructura más detallada y granularidad en las características de red. La aplicación de técnicas avanzadas de codificación y normalización asegura que el modelo pueda capturar efectivamente la naturaleza jerárquica y secuencial del tráfico de red, lo que es esencial para detectar patrones de ataque en un entorno de red realista.

3.4. Entrenamiento del modelo

Dado que las características de Litnet difieren significativamente de los datos sintéticos originales, el modelo se entrena desde cero. Esto permite al modelo aprender características relevantes directamente de los datos realistas de Litnet, maximizando la efectividad de la transferencia de arquitectura.

Configuración del Entrenamiento:

- Se utiliza un tamaño de lote adecuado, un número suficiente de épocas y técnicas de regularización como Dropout para evitar el sobreajuste durante el entrenamiento.

- La función de pérdida implementada es la EQL v2 (*Equalization Loss v2*), que ajusta dinámicamente el peso de las clases minoritarias, lo cual es esencial para manejar el desbalance de clases en Litnet.

3.4.1. Definición de hiper parámetros

El modelo propuesto, basado en la arquitectura de CANET, se entrena desde cero utilizando el conjunto de datos Litnet. Para optimizar el rendimiento del modelo, se definieron los siguientes hiperparámetros:

- Dispositivo de entrenamiento: El modelo se entrena en GPU (CUDA) si está disponible; de lo contrario, se utiliza la CPU.
- Modelo: La arquitectura base del modelo es una red convolucional con atención jerárquica, adaptada a las características específicas de Litnet.
- Función de pérdida: Se implementa la función de pérdida EQL v2 (*Equalization Loss v2*) para manejar el desequilibrio de clases en Litnet. Esta función ajusta dinámicamente los pesos de las clases minoritarias durante el entrenamiento, mejorando la capacidad del modelo para identificar clases menos representadas.
- Optimizador: Se utiliza el optimizador Adam con una tasa de aprendizaje inicial de 0.001, elegido por su eficacia en la optimización de modelos de redes neuronales profundas.
- Tamaño de lote: Se establece un tamaño de lote de 256 para garantizar un equilibrio entre la estabilidad del entrenamiento y la eficiencia computacional.
- Número de épocas: Se establecen 10 épocas de entrenamiento, suficientes para asegurar la convergencia del modelo y la estabilidad en la precisión.
- Validación cruzada con K-Fold: Se aplica validación cruzada con K-Fold estratificada (n=5), lo que permite evaluar la consistencia del modelo en diferentes particiones del conjunto de datos y mejora la robustez del entrenamiento.

3.4.2. Configuración del entrenamiento

El proceso de entrenamiento se estructura en varias etapas para garantizar un aprendizaje eficiente y monitorear el desempeño del modelo durante cada iteración. La configuración incluye:

- División de los datos: Se utiliza validación cruzada estratificada con 5 pliegues, lo que implica dividir el conjunto de datos Litnet en partes de entrenamiento y prueba en cada pliegue. Esta técnica garantiza que cada pliegue mantenga la proporción original de clases, mejorando la estabilidad del modelo en presencia de datos desbalanceados. Los conjuntos de entrenamiento y prueba se generan para cada pliegue, lo que permite evaluar el rendimiento del modelo en diferentes subconjuntos de Litnet.
- Inicialización del modelo: Para cada pliegue, se inicializa el modelo desde cero, asegurando que el entrenamiento sea independiente y que el modelo no esté sesgado por iteraciones previas.
- Carga de datos: Los datos de entrenamiento se cargan mediante un DataLoader que permite la iteración por lotes en el conjunto de datos, lo que asegura una mayor eficiencia en la carga de datos y en la utilización de recursos de hardware.
- Entrenamiento por épocas: Para cada época, se ejecuta un bucle de entrenamiento en el que se realiza la propagación hacia adelante y hacia atrás. Durante cada lote de entrenamiento, se calculan las predicciones, se evalúa la función de pérdida y se actualizan los pesos del modelo a través del optimizador Adam. Al final de cada época, se calcula la pérdida media y la precisión del modelo en el conjunto de entrenamiento, y se almacenan para su análisis posterior.

3.4.3. Monitorización del rendimiento

La monitorización del rendimiento es un aspecto clave del entrenamiento del modelo, ya que permite evaluar la efectividad del proceso y ajustar hiperparámetros en función de los resultados obtenidos. La monitorización se realiza mediante las siguientes métricas:

- Pérdida por época: Se calcula la pérdida media al final de cada época para evaluar la estabilidad del modelo durante el entrenamiento. La pérdida se almacena para cada pliegue y

se grafica para visualizar la evolución del entrenamiento.

- Precisión por época: Se calcula la precisión al final de cada época para medir la proporción de predicciones correctas en el conjunto de entrenamiento. Esta métrica se utiliza para evaluar la capacidad del modelo para aprender características relevantes de Litnet.
- Pesos de las clases: Durante el entrenamiento, se monitorea la evolución de los pesos de las clases en la función de pérdida EQL v2, lo que permite evaluar cómo se ajustan dinámicamente para mejorar el aprendizaje en clases minoritarias. La evolución de los pesos se grafica para visualizar su comportamiento a lo largo de las iteraciones.
- Evaluación final por pliegue: Al final de cada pliegue, se evalúa el modelo en el conjunto de prueba correspondiente. Se calculan métricas como precisión, tasa de detección (recall), tasa de falsos positivos (FPR) y F1-score para evaluar el rendimiento del modelo en cada partición de Litnet.
- Curvas de Pérdida y Precisión: Se grafican las curvas de pérdida y precisión para cada pliegue, lo que permite identificar patrones de convergencia y posibles problemas de sobre-ajuste o subajuste.
- Matriz de Confusión: Se construye una matriz de confusión para cada pliegue y se visualiza para analizar la distribución de errores y el desempeño del modelo en cada clase. Esto permite identificar áreas de mejora en la clasificación de clases específicas.

3.4.4. Entrenamiento del Modelo

El modelo convolucional con atención jerárquica fue entrenado desde cero utilizando los datos preprocesados de Litnet. Se utilizó una estrategia de validación cruzada con `StratifiedKFold` para asegurar la generalización del modelo y la correcta detección de clases desbalanceadas. El proceso de entrenamiento involucró:

- **Configuración del Entrenamiento:** El modelo se entrenó con `Adam` como optimizador y una tasa de aprendizaje de 0.001. La función de pérdida elegida fue `EQLv2`, una función de

pérdida diseñada para mejorar la detección de clases minoritarias, lo cual es crítico para un conjunto de datos desbalanceado como Litnet.

- **Monitoreo de Gradientes:** Durante el entrenamiento, se implementó un monitoreo de gradientes positivo y negativo acumulado por clase en cada época para ajustar dinámicamente los pesos de la función de pérdida. Esto permitió mejorar la detección de intrusos al dar mayor importancia a las clases minoritarias.
- **Uso de Atención Jerárquica:** La atención jerárquica se implementó en tres niveles de la red convolucional para identificar las características más relevantes en la detección de cada tipo de ataque. Este mecanismo mejoró la interpretación del modelo al proporcionar una comprensión más clara de qué características temporales y de flujo son más críticas para la detección.
- **Ajustes del Modelo:** Después de completar el entrenamiento, se realizaron ajustes de hiperparámetros y análisis de desempeño para optimizar la arquitectura y la función de pérdida. La evaluación final se realizó utilizando métricas de precisión, tasa de detección (recall), tasa de falsos positivos (FPR), y el puntaje F1, asegurando un análisis completo de la eficacia del modelo.

3.4.5. Estrategia de entrenamiento

La estrategia de entrenamiento varió en varios aspectos:

- **Estrategia:**
  - El modelo base se utilizo entrenamiento estratificado de 10 pliegues lo que hace lento.
  - Modelo propuesto fue entrenado utilizando 70 para entrenamiento, 15 para pruebas y 15 para validación se utilizaron técnicas de carga de datos más eficientes, como el uso de *DataLoader* de PyTorch, lo que permitió un manejo más eficiente de grandes volúmenes de datos
- **Optimización:**
  - **Modelo original:** El optimizador Adam fue utilizado con una tasa de aprendizaje fija.

- **Modelo ajustado:** Se utilizó Adam con una tasa de aprendizaje más baja, lo cual se ajustó dinámicamente durante el entrenamiento para una mejor convergencia.

- **Monitoreo de gradientes:**
  - En el modelo ajustado, se implementó un monitoreo de gradientes positivos y negativos al inicio y al final de cada época, lo que permitió un control más preciso de los ajustes en la ponderación de las clases, adaptándose mejor a la dinámica del desbalance de clases.

3.4.6. Fórmulas matemáticas

Definición de la pérdida durante el entrenamiento:

$$\mathcal{L}_{\text{train}} = \frac{1}{|B|} \sum_{(x_i, y_i) \in B} \text{EQLv2}(f(x_i; \theta), y_i)$$

Donde:

- EQLv2: Función de pérdida para datos desbalanceados.
- $f(x_i; \theta)$: Predicción del modelo con parámetros θ.
- y_i: Etiqueta verdadera para la muestra x_i.
- B: Lote de datos.

Actualización de los parámetros del modelo:

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}_{\text{train}}}{\partial \theta}$$

Donde:

- η: Tasa de aprendizaje.
- $\frac{\partial \mathcal{L}_{\text{train}}}{\partial \theta}$: Gradiente de la pérdida con respecto a los parámetros θ.

Cálculo de la precisión:

$$\text{Precisión} = \frac{\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i)}{N}$$

Donde:

- $\mathbb{1}$: Función indicadora que es 1 si $\hat{y}_i = y_i$, y 0 en caso contrario.
- N: Número total de muestras.

Criterio de early stopping:

$$\text{Si } \mathcal{L}_{\text{val}}^{(t)} \geq \mathcal{L}_{\text{val}}^{\text{best}}, \quad \text{incrementar } p_{\text{no\_improve}}$$

$$\text{Si } p_{\text{no\_improve}} \geq \text{patience}, \quad \text{detener el entrenamiento.}$$

3.5. Ajuste del modelo

3.5.1. Ajuste de hiper parámetros

3.5.2. Optimización adicional

Detalle de cualquier técnica de optimización utilizada después del entrenamiento inicial, como regularización (dropout, L2), aumento de datos (data augmentation), o ajustes de tasa de aprendizaje. Evaluación del rendimiento del modelo durante el ajuste fino, con énfasis en la estabilidad y la convergencia del modelo.

3.5.3. Evaluación iterativa

Descripción de las iteraciones del modelo para mejorar el rendimiento en clases minoritarias y reducir la tasa de falsos positivos, ajustando la función de pérdida o el mecanismo de atención según sea necesario.

Algorithm 3 Estrategia de entrenamiento con pérdida EQLv2 y early stopping

Entrada: Datos de entrenamiento D_{train}, datos de validación D_{val}, modelo $f(x;\theta)$, número de épocas T, paciencia p.
Salida: Modelo entrenado $f(x;\theta)$.
Inicializar parámetros del modelo θ;
Inicializar optimizador Adam con tasa de aprendizaje η;
Inicializar la mejor pérdida de validación $\mathcal{L}_{\text{val}}^{\text{best}} \leftarrow \infty$;
Inicializar contador de paciencia $p_{\text{no\_improve}} \leftarrow 0$;
for $t \leftarrow 1$ **to** T **do** ▷ Iterar sobre cada época
    Fase de Entrenamiento:
    Establecer el modelo en modo entrenamiento;
    Calcular la pérdida de entrenamiento $\mathcal{L}_{\text{train}}$ y actualizar θ usando Adam;
    Fase de Validación:
    Establecer el modelo en modo evaluación;
    Calcular la pérdida de validación $\mathcal{L}_{\text{val}}$ y las métricas (Precisión, Recall, F1-Score);
    if $\mathcal{L}_{\text{val}} < \mathcal{L}_{\text{val}}^{\text{best}}$ **then**
        $\mathcal{L}_{\text{val}}^{\text{best}} \leftarrow \mathcal{L}_{\text{val}}$;
        Guardar los parámetros del modelo θ;
        $p_{\text{no\_improve}} \leftarrow 0$;
    else
        $p_{\text{no\_improve}} \leftarrow p_{\text{no\_improve}} + 1$;
    end if
    if $p_{\text{no\_improve}} \geq p$ **then**
        Detener entrenamiento.
        Salir del bucle.
        break;
    end if
end for
Devolver: Modelo entrenado $f(x;\theta)$.

3.6. Evaluación del modelo

3.6.1. Evaluación del modelo

La evaluación se llevó a cabo utilizando la matriz de confusión y el informe de clasificación para cada pliegue, lo cual permitió identificar la efectividad del modelo en la detección de diferentes tipos de ataques. La precisión, la tasa de detección, y la tasa de falsos positivos fueron reportadas para cada clase, permitiendo una evaluación detallada del desempeño del modelo.

3.6.2. Métricas de rendimiento

Para evaluar el desempeño del modelo, se utilizan varias métricas que permiten medir tanto la efectividad global como el rendimiento por clase:

Precisión (Accuracy): Mide la proporción de predicciones correctas con respecto al total de muestras, reflejando la capacidad del modelo para clasificar correctamente las intrusiones y el tráfico normal. Tasa de Detección (Recall/Detection Rate): Calcula la proporción de instancias de intrusiones detectadas correctamente con respecto al total de intrusiones reales. Es una métrica crítica, ya que refleja la sensibilidad del modelo en la identificación de intrusiones. Tasa de Falsos Positivos (FPR): Indica la proporción de muestras normales clasificadas incorrectamente como intrusiones, lo que permite evaluar la confiabilidad del modelo en la minimización de alarmas falsas. F1-Score: Combina precisión y tasa de detección en una métrica equilibrada, proporcionando una visión más completa del rendimiento del modelo, especialmente en clases desbalanceadas.

- **Precisión (*Accuracy*):** La proporción de predicciones correctas sobre el total de predicciones. Se evaluó tanto en el conjunto de validación como en el de prueba para asegurar que el modelo mantenía un buen rendimiento en datos no vistos.
- **Tasa de Detección (*Recall*):** La capacidad del modelo para identificar correctamente los ataques en el tráfico de red. Un alto *recall* es crítico en la detección de intrusos, ya que permite minimizar los ataques no detectados.
- **Tasa de Falsos Positivos:** El porcentaje de predicciones incorrectas en las que el modelo identificó tráfico normal como malicioso. Se evaluó la tasa de falsos positivos para garantizar

que el modelo no genere demasiadas alertas innecesarias, lo que podría reducir su utilidad práctica.

- **F1-Score:**Combina precisión y tasa de detección en una métrica equilibrada, proporcionando una visión más completa del rendimiento del modelo, especialmente en clases desbalanceadas.

3.6.3. Análisis de resultados por clase

El análisis de resultados por clase es esencial para identificar cómo el modelo maneja las diferentes clases de tráfico en Litnet, especialmente considerando la naturaleza desbalanceada del conjunto de datos.

- **Clasificación por clase:**

  Se presenta un informe de clasificación para cada pliegue, que incluye precisión, tasa de detección y F1-Score por clase. Esto permite evaluar el rendimiento del modelo en la identificación de diferentes tipos de intrusiones, como 'http_f', 'icmp_f', 'smtp_b', entre otros.

  La matriz de confusión generada para cada pliegue proporciona una representación visual de los aciertos y errores de clasificación por clase. La matriz de confusión final se presenta al completar la validación cruzada, mostrando la distribución de predicciones correctas e incorrectas por clase.

- **Observaciones claves:**

  Se destacan las clases con alta tasa de detección y aquellas con dificultades para ser clasificadas correctamente.

  Se analiza cómo el modelo maneja las clases minoritarias y si la implementación de la función de pérdida EQL v2 mejora la detección de estas clases.

3.6.4. Análisis de errores

El análisis de errores permite identificar las limitaciones del modelo y las posibles áreas de mejora. Se enfoca en los siguientes aspectos:

- Tasa de Falsos Positivos

Se analiza la tasa de falsos positivos (FPR) para determinar qué tipos de tráfico normal se clasifican incorrectamente como intrusiones. Este análisis es clave para ajustar el modelo y reducir las alarmas falsas, mejorando la confiabilidad del SDI. Se evalúan las clases que contribuyen más a los falsos positivos, lo que permite ajustar el umbral de decisión o realizar ajustes adicionales en el mecanismo de atención para reducir estos errores.

- Falsos Negativos:

  Se identifican los casos en los que el modelo no logra detectar una intrusión, lo que representa un riesgo significativo en la efectividad del SDI. Se analizan las características de las intrusiones no detectadas para determinar si el modelo necesita una mayor adaptación en ciertas clases o características de tráfico.

- Visualización de Pesos de Clase:

  Se presenta un gráfico de la evolución de los pesos de clase durante el entrenamiento, lo que permite visualizar cómo el modelo ajusta su enfoque hacia clases minoritarias a medida que avanza el entrenamiento. Este análisis ayuda a entender el comportamiento del modelo en términos de priorización de clases, lo que es fundamental para mejorar la detección de intrusiones menos frecuentes.

Fórmulas matemáticas para la evaluación del modelo

Cálculo de la pérdida en el conjunto de prueba:

$$\mathcal{L}_{\text{test}} = \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} \text{Pérdida}(f(x_i;\theta), y_i)$$

Donde:

- Pérdida$(f(x_i;\theta), y_i)$ es la función de pérdida calculada para la predicción $f(x_i;\theta)$ y la etiqueta real y_i.
- N_{total} es el número total de muestras en el conjunto de prueba.

Cálculo de la precisión:

$$\text{Precisión} = \frac{\sum_{i=1}^{N_{\text{total}}} \mathbb{1}(\hat{y}_i = y_i)}{N_{\text{total}}}$$

Donde:

- $\mathbb{1}$ es la función indicadora que es 1 si $\hat{y}_i = y_i$, y 0 en caso contrario.

Cálculo de la tasa de detección (recall):

$$\text{DR (Recall)} = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} (TP_c + FN_c)}$$

Donde:

- TP_c es el número de verdaderos positivos para la clase c.
- FN_c es el número de falsos negativos para la clase c.

Cálculo de la tasa de falsos positivos (FPR):

$$\text{FPR} = \frac{\sum_{c=1}^{C} FP_c}{\sum_{c=1}^{C} (FP_c + TN_c)}$$

Donde:

- FP_c es el número de falsos positivos para la clase c.
- TN_c es el número de verdaderos negativos para la clase c.

Cálculo de la puntuación F1 (F1-Score):

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}}$$

Fórmulas matemáticas para analizar el impacto de la clase mayoritaria

Distribución de clases en el conjunto de prueba:

$$\text{Distribución de clases} = \{\text{clase}_i : \text{frecuencia}_i\}$$

Algorithm 4 Evaluación del modelo en el conjunto de prueba

Entrada: Modelo entrenado $f(x;\theta)$, datos de prueba D_{test}, función de pérdida criterion.
Salida: Métricas de evaluación: Precisión, Recall, F1-Score, Tasa de Falsos Positivos (FPR).
Cargar el mejor modelo guardado por Early Stopping.
Inicializar acumuladores para las métricas: $test\_correct\_predictions = 0$, $test\_total\_predictions = 0$, $test\_loss = 0$.
Inicializar listas vacías y_pred y y_true para las predicciones y etiquetas reales.
for cada lote de datos en D_{test} **do**
    Predecir las salidas con $outputs = f(inputs)$.
    Calcular la pérdida en el conjunto de prueba: $test\_loss+ =$ criterion$(outputs, targets)$.
    Contar predicciones correctas: $test\_correct\_predictions+ = \sum(\hat{y}_i = y_i)$.
    Actualizar el número total de predicciones: $test\_total\_predictions+ = N$.
    Guardar las predicciones y las etiquetas verdaderas en las listas y_pred y y_true.
end for
Calcular la precisión: Precisión $= \frac{test\_correct\_predictions}{test\_total\_predictions}$.
Calcular el recall, la tasa de falsos positivos (FPR) y el F1-Score.
Mostrar las métricas de evaluación.
Calcular la matriz de confusión y graficar.

Donde frecuencia$_i$ es el número de instancias de la clase i en el conjunto de prueba.

Cálculo de la tasa de falsos positivos (FPR) por clase:

$$\text{FPR}_{\text{clase}_c} = \frac{FP_c}{FP_c + TN_c}$$

Donde:

- FP_c es el número de falsos positivos para la clase c.
- TN_c es el número de verdaderos negativos para la clase c.

Cálculo de la clase mayoritaria:

$$\text{Clase mayoritaria} = \arg\max(\text{Distribución de clases})$$

Esto nos da la clase con la mayor cantidad de instancias en el conjunto de prueba.

Filtrado de la clase mayoritaria:

$$y_{\text{true\_filtered}} = \{y_i \mid y_i \neq \text{Clase mayoritaria}\}$$

$$y_{\text{pred\_filtered}} = \{y_i \mid y_i \neq \text{Clase mayoritaria}, \hat{y}_i\}$$

Cálculo de la matriz de confusión filtrada:

$$\text{cm\_filtered} = \text{confusion\_matrix}(y_{\text{true\_filtered}}, y_{\text{pred\_filtered}})$$

Esto nos da la matriz de confusión sin la clase mayoritaria.

Cálculo del FPR ponderado:

$$\text{FPR ponderada} = \sum_{c=1}^{C} \text{FPR}_{\text{clase}_c} \cdot \frac{\text{frecuencia}_c}{\sum_{i=1}^{C} \text{frecuencia}_i}$$

Donde:

- frecuencia_c es el número de instancias de la clase c en el conjunto de prueba.

Cálculo del FPR sin la clase mayoritaria:

$$\text{FPR sin clase mayoritaria} = \frac{\sum_{c=1}^{C} FP_{\text{filtered}}}{\sum_{c=1}^{C} (FP_{\text{filtered}} + TN_{\text{filtered}})}$$

Esto nos da el FPR sin la clase mayoritaria.

Capítulo 4

Experimentos y Resultados

4.1. Análisis de los resultados del modelo

4.1.1. Resultados del modelo evaluado en Litnet

Este capítulo presenta los resultados obtenidos por el modelo evaluado con el conjunto de datos Litnet, destacando métricas esenciales como precisión, tasa de detección (DR) y tasa de falsos positivos (FPR). A continuación, se realiza una comparación con los resultados de CANet, previamente evaluado en datos sintéticos.

4.1.2. Resultados generales del modelo

Los resultados globales alcanzados por el modelo en el conjunto Litnet muestran una precisión y una capacidad de detección casi perfectas, como se observa en la Tabla 4.1.

Tabla 4.1. Resultados Generales del Modelo en Litnet

Métrica	Valor
Precisión Global (Accuracy)	0.9994
Tasa de Detección (Recall)	0.9994
Tasa de Falsos Positivos (FPR)	0.0000
Puntuación F1 Global	0.9994

4.1.3. Desempeño por clase

En la Tabla 4.2 se detalla el desempeño del modelo en términos de precisión, recall y F1-score para cada clase individual del conjunto de datos Litnet, lo cual proporciona una visión detallada de la eficacia del modelo en diferentes tipos de tráfico.

Tabla 4.2. Desempeño por clase en Litnet

Clase	Precisión	Recall	F1-Score	Soporte
http_f	0.95	1.00	0.97	77
icmp_f	0.93	1.00	0.96	68
icmp_smf	1.00	1.00	1.00	367
Normal	1.00	1.00	1.00	109212
smtp_b	1.00	0.50	0.67	2
tcp_land	0.98	1.00	0.99	157
tcp_red_w	1.00	1.00	1.00	3774
tcp_syn_f	1.00	1.00	1.00	4771
tcp_udp_win_p	1.00	1.00	1.00	19
tcp_w32_w	0.97	0.75	0.84	75
udp_0	1.00	0.50	0.67	2
udp_f	0.97	0.95	0.96	282
udp_reaper_w	1.00	1.00	1.00	5

4.1.4. Matriz de confusión

Para ilustrar visualmente la distribución de los aciertos y errores, se presenta la matriz de confusión en la Figura 4.1, donde se puede observar la efectividad del modelo en identificar correctamente las etiquetas de cada clase.

4.2. Análisis detallado de resultados

Los resultados obtenidos en el conjunto de datos Litnet con la arquitectura propuesta muestran un desempeño notable en cuanto a precisión, tasa de detección y tasa de falsos positivos, como se evidencia en la Tabla 4.1. A continuación, se proporciona un análisis exhaustivo del rendimiento del modelo a nivel de clase y se compara su eficacia con el modelo CANet, evaluado en conjuntos de datos sintéticos.

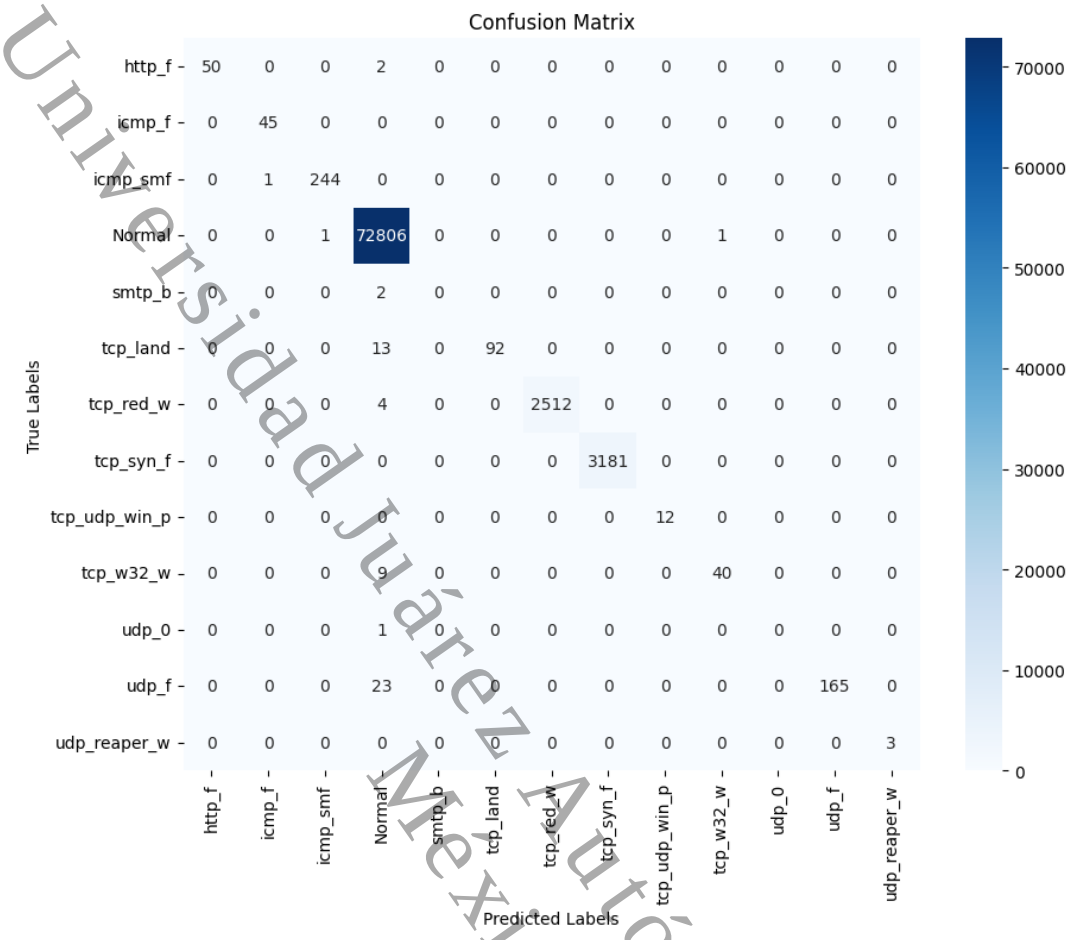


Figura 4.1. Matriz de Confusión del Modelo en Litnet

4.2.1. Análisis global

En términos globales, el modelo evaluado en Litnet alcanzó una **precisión de 99.94 %** y una **tasa de detección de 99.94 %**, sin generar falsos positivos (FPR de 0.00 %). Este rendimiento refleja una capacidad robusta del modelo para identificar correctamente patrones de tráfico tanto normales como anómalos en un entorno de datos reales. La puntuación F1 de 99.94 % confirma que el modelo mantiene un equilibrio óptimo entre precisión y recall, esencial para aplicaciones de detección de intrusiones donde tanto la identificación correcta como la minimización de falsos positivos son críticas.

4.2.2. Análisis por clase

La Tabla 4.2 resume el desempeño del modelo para cada clase de tráfico. El **tráfico normal** representa la mayor proporción de datos, y el modelo logra identificarlo casi sin errores (precisión y recall de 100 %). Las clases *icmp_smf*, *tcp_red_w*, y *udp_reaper_w* también alcanzan puntuaciones F1 de 100 %, indicando que el modelo es altamente preciso en la detección de estas categorías.

Por otro lado, algunas clases con soporte bajo, como *smtp_b* y *udp_0*, presentan desafíos en términos de recall y precisión. Para la clase *smtp_b*, por ejemplo, el modelo obtiene un recall de 50 %, lo que implica que la mitad de las instancias de esta clase no son detectadas correctamente. Este fenómeno puede atribuirse a la baja frecuencia de estas categorías en el conjunto de datos, lo que limita la capacidad del modelo para aprender patrones distintivos. A pesar de estos retos, la alta precisión en las clases comunes es indicativa de la efectividad del modelo en la mayoría de los casos de tráfico de red.

4.2.3. Interpretación y discusión de los resultados

La excelente precisión y recall alcanzados por el modelo en Litnet validan la arquitectura con atención jerárquica multihead y normalización añadida tras cada capa de atención. Estos elementos contribuyen a que el modelo se adapte eficazmente a la diversidad y complejidad del tráfico real de red, mejorando tanto la capacidad de generalización como la sensibilidad en la detección de intrusiones.

La comparación con CANet sugiere que, aunque los modelos entrenados en datos sintéticos son útiles para evaluar arquitecturas, los resultados obtenidos con Litnet demuestran una mayor aplicabilidad en entornos reales. La baja tasa de falsos positivos es particularmente significativa, ya que indica una menor probabilidad de alarmas incorrectas en un sistema de detección de intrusos implementado, lo que mejora la confiabilidad del sistema.

Los resultados obtenidos en Litnet confirman que el modelo propuesto es altamente eficaz para la detección de intrusiones en redes reales. Su rendimiento destaca en comparación con modelos previos evaluados en datos sintéticos, posicionándolo como una opción robusta y confiable para aplicaciones de ciberseguridad en tiempo real. Las mejoras en las clases con bajo soporte podrían abordarse mediante técnicas de aumento de datos o ponderación de clases, lo

que constituye una posible línea de investigación futura.

4.3. Evaluación de la función de perdida del modelo

En esta sección se presentan los resultados obtenidos al entrenar el modelo H.A.L.C.C.O.N utilizando datos reales de tráfico de red (Litnet). Para este experimento, se utilizó la función de pérdida EQLv2 con un ajuste de los gradientes dinámicos basado en las clases raras, incluyendo el parámetro $\mu = 0.8$, con el objetivo de analizar su impacto en la estabilidad y efectividad del modelo.

4.3.1. Métricas globales

El modelo alcanzó un rendimiento sobresaliente con las siguientes métricas globales:

- **Train Accuracy**: 0.9994
- **Validation Accuracy**: 0.9996
- **F1-Score Final**: 0.9995

Estas métricas destacan la capacidad del modelo para abordar un escenario con un alto desbalance de clases.

4.3.2. Análisis de gradientes acumulados

El comportamiento de los gradientes acumulados, tanto positivos como negativos, fue monitoreado durante las 6 épocas del entrenamiento. Las gráficas a continuación muestran la evolución de los gradientes por clase.

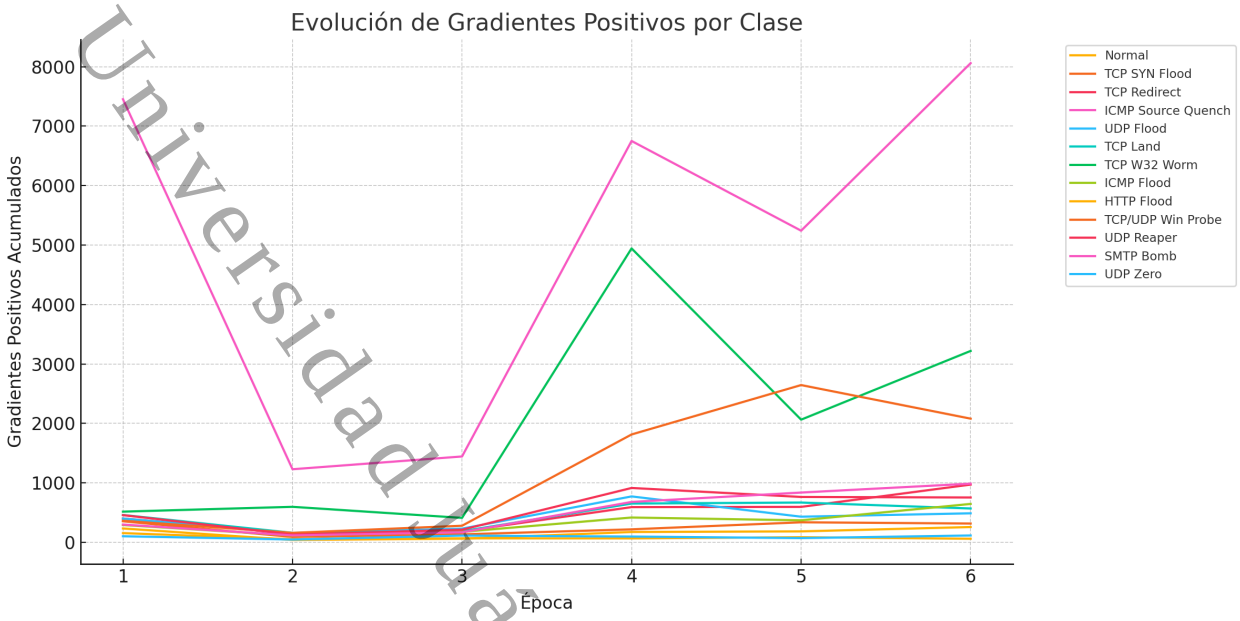


Figura 4.2. Evolución de los gradientes positivos acumulados por clase durante el entrenamiento.

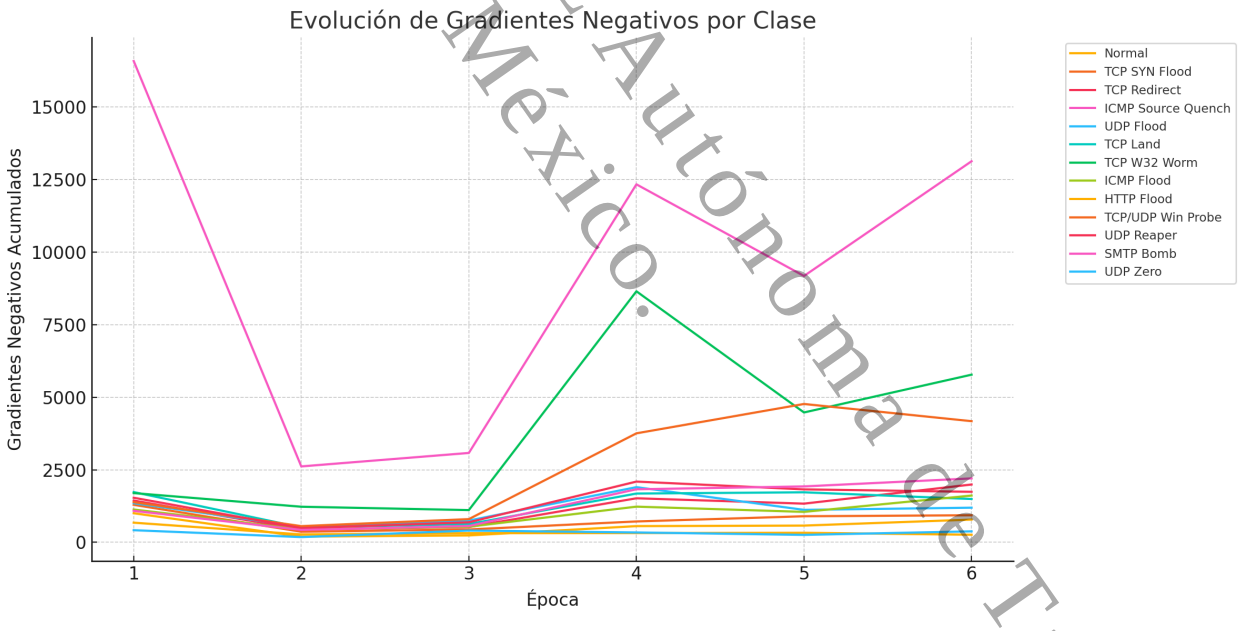


Figura 4.3. Evolución de los gradientes negativos acumulados por clase durante el entrenamiento.

Las clases raras, como *SMTP Bomb* y *UDP Zero*, muestran un incremento progresivo en sus gradientes positivos, lo que refleja el ajuste dinámico de la función de pérdida EQLv2.

4.3.3. Distribución de clases en los datos reales

Para comprender el impacto del desbalance en los resultados, a continuación, se presenta la distribución de clases en el conjunto de datos reales utilizado para el entrenamiento.

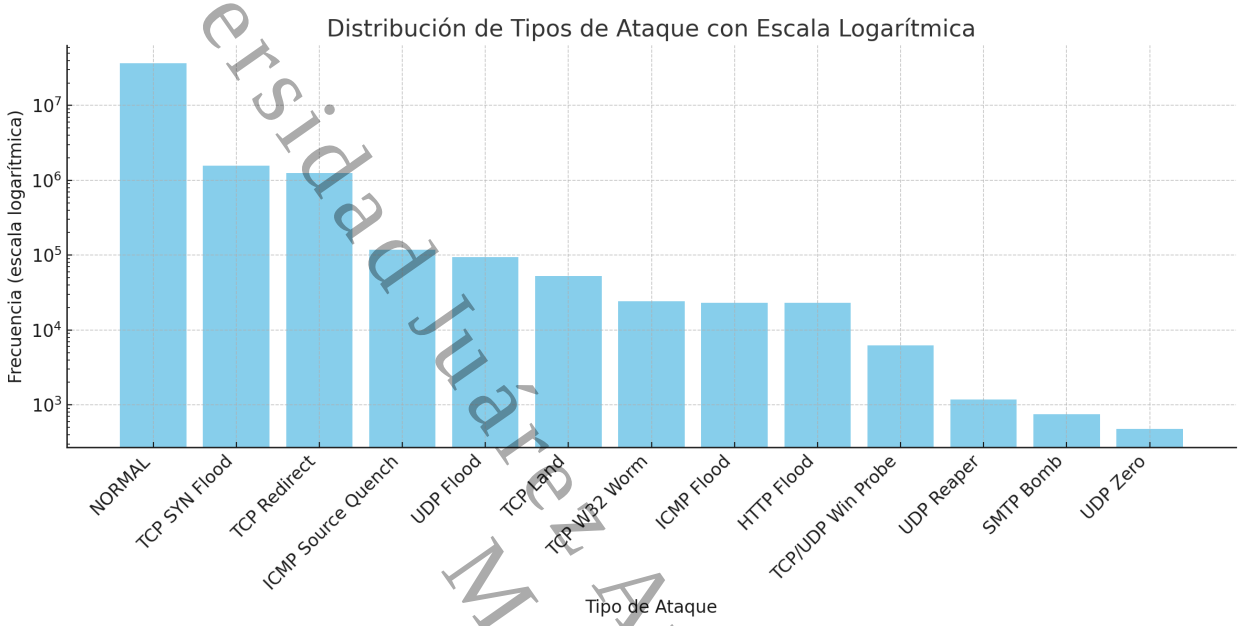


Figura 4.4. Distribución de clases en escala logarítmica en el conjunto de datos Litnet.

La clase *Normal* domina el conjunto de datos con más del 90 % de las muestras, mientras que las clases raras, como *SMTP Bomb* y *UDP Zero*, representan menos del 0.01 %.

4.3.4. Impacto del uso de EQLv2

El uso de EQLv2 permitió:

- Ajustar dinámicamente los pesos de las clases raras, aumentando su contribución al cálculo de la pérdida.
- Mantener la estabilidad del modelo incluso en presencia de un desbalance extremo.
- Mejorar el recall y el F1-Score en clases con baja representación.

4.4. Comparación con otros modelos

En esta sección se realiza una comparación de los resultados obtenidos con el modelo H.A.L.C.C.O.N y los resultados del modelo CANET, evaluado con datos sintéticos. Los resultados comparativos se resumen en la Tabla 4.3.

Tabla 4.3. Comparación entre H.A.L.C.C.O.N (Litnet) y CANET (NSL-KDD y UNSW-NB15).

Métrica	H.A.L.C.C.O.N (Litnet)	CANET (NSL-KDD)	CANET (UNSW-NB15)
Accuracy (%)	99.94	99.67	87.53
Detection Rate (%)	99.95	99.68	97.84
False Positive Rate (%)	0.0036	0.32	2.66

Los resultados indican que H.A.L.C.C.O.N supera a los modelos estado de arte en escenarios de datos reales debido a la capacidad de EQLv2 para manejar desbalances extremos.

El modelo H.A.L.C.C.O.N, utilizando la pérdida EQLv2, demuestra ser efectivo para abordar el desbalance de clases en datos reales. Los ajustes realizados en los gradientes y la dinámica de los pesos permitieron alcanzar un F1-Score sobresaliente de 0.9995, justificando el uso de esta arquitectura para sistemas de detección de intrusos basados en aprendizaje profundo.

4.5. Comparación con otros modelos estado del arte en datos sintéticos

En la Tabla 4.4, se compara el rendimiento de nuestro modelo en Litnet con el modelo CANet evaluado en conjuntos de datos sintéticos NSL-KDD y UNSW-NB15. Estos resultados destacan la eficacia de nuestra arquitectura en un entorno de datos reales.

Tabla 4.4. Comparación del Desempeño de Litnet vs. CANet en Datos Sintéticos

Modelo	Conjunto	Exactitud (Accuracy)	Tasa de Detección (DR)	FPR
CANet	NSL-KDD	99.67 %	99.68 %	0.32 %
CANet	UNSW-NB15	87.53 %	97.84 %	2.66 %
Litnet	Litnet	99.94 %	99.94 %	0.00 %

El modelo base alcanza una precisión promedio de 99.67 % en NSL-KDD y 87.53 % en UNSW-

NB15. Estas métricas, si bien son sólidas, quedan por debajo de la precisión del modelo propuesto en Litnet, especialmente cuando se considera la tasa de falsos positivos (2.66 % en UNSW-NB15 frente a 0 % en Litnet). Este resultado sugiere que, al evaluar en datos reales de red, el modelo propuesto tiene una capacidad superior de generalización y robustez frente a variaciones en el tráfico, minimizando los falsos positivos y manteniendo una alta precisión.

4.6. Comparación con otros modelo de aprendizaje automático

4.6.1. Experimento con regresión logística multinomial

La regresión logística multinomial se seleccionó para este experimento debido a su capacidad para manejar problemas de clasificación multiclase, lo cual es relevante para la detección de intrusos en este trabajo. Este modelo permite obtener probabilidades de pertenencia a clases, lo que facilita la interpretación y análisis del comportamiento de las diferentes clases de ataques.

4.6.1.1. Configuración del modelo

Para este experimento, se utilizó la implementación de `LogisticRegression` de la librería `scikit-learn`. El modelo se configuró en modo multinomial, utilizando el solver `lbfgs` y estableciendo un máximo de 500 iteraciones para asegurar su convergencia. La configuración se dividió en `X_train` y `y_train` para el entrenamiento, y `X_test` y `y_test` para la evaluación.

4.6.1.2. Resultados de las predicciones

El reporte de clasificación obtenido del modelo de regresión logística multinomial se presenta en la Tabla 4.5. Las métricas de precisión, recall y F1-score para cada clase del conjunto de datos se resumen de la siguiente manera:

Tabla 4.5. Reporte de Clasificación del Modelo de Regresión Logística Multinomial

Clase	Precisión	Recall	F1-score	Soporte
http_f	0.00	0.00	0.00	98
icmp_f	0.00	0.00	0.00	100
icmp_smf	0.88	0.98	0.93	507
Normal	0.96	0.99	0.98	145608
smtp_b	0.00	0.00	0.00	1
tcp_land	0.00	0.00	0.00	206
tcp_red_w	0.00	0.00	0.00	4940
tcp_syn_f	0.88	1.00	0.94	6424
tcp_udp_win_p	1.00	1.00	1.00	21
tcp_w32_w	0.00	0.00	0.00	97
udp_0	0.00	0.00	0.00	1
udp_f	0.27	0.10	0.14	406
udp_reaper_w	0.00	0.00	0.00	6
Exactitud		0.96		158415
Promedio Macro	0.31	0.31	0.31	-
Promedio Ponderado	0.92	0.96	0.94	-

4.6.1.3. Validación cruzada

Para evaluar la estabilidad del modelo, se realizó una validación cruzada utilizando 5 folds. La precisión promedio obtenida durante esta validación fue de 0.9025, lo que indica un desempeño robusto del modelo en los diferentes subconjuntos del conjunto de datos.

4.6.1.4. Interpretación de los resultados

Los resultados del modelo de regresión logística multinomial muestran un desempeño satisfactorio en las clases mayoritarias (por ejemplo, Clase 3), donde la precisión y el recall son altos, lo que contribuye significativamente a la precisión general del modelo (96%). Sin embargo, se observa un desempeño deficiente en las clases minoritarias, como las Clases 0, 1, 4, 5, 6, 9, 10 y 12, lo cual es coherente con la naturaleza desbalanceada del conjunto de datos de entrenamiento. Estos resultados reflejan la necesidad de técnicas adicionales de balanceo de clases o ajuste de hiperparámetros para mejorar la capacidad del modelo de clasificar correctamente las clases minoritarias.

En resumen, el experimento con la regresión logística multinomial proporciona una línea base de rendimiento para la detección de intrusos en este conjunto de datos. Si bien el modelo muestra

limitaciones en la clasificación de clases minoritarias, ofrece una perspectiva inicial de las mejoras potenciales que se pueden implementar en modelos más complejos.

4.6.2. Experimento con K-Nearest Neighbors (KNN)

El algoritmo K-Nearest Neighbors (KNN) es un método de clasificación supervisada ampliamente utilizado en problemas de clasificación multiclase. En este experimento, se evaluó su desempeño en el contexto de la detección de intrusos, aprovechando su simplicidad y eficacia para manejar grandes conjuntos de datos como el utilizado en este estudio.

4.6.2.1. Configuración del modelo

Para este experimento, se utilizó la implementación de `KNeighborsClassifier` de la librería `scikit-learn`. El modelo se configuró con 5 vecinos (`n_neighbors=5`), lo que permitió evaluar la proximidad de cada muestra a sus vecinos más cercanos para la clasificación. Se dividieron los datos en conjuntos de entrenamiento y prueba, representados por `X_train`, `y_train`, `X_test` y `y_test`, respectivamente.

4.6.2.2. Resultados de las predicciones

El reporte de clasificación obtenido del modelo KNN se presenta en la Tabla 4.6. Las métricas de precisión, recall y F1-score para cada clase se resumen a continuación:

Tabla 4.6. Reporte de clasificación del modelo KNN

Clase	Precisión	Recall	F1-score	Soporte
http_f	0.31	0.08	0.13	98
icmp_f	0.81	0.77	0.79	100
icmp_smf	0.98	1.00	0.99	507
Normal	0.98	0.98	0.98	145608
smtp_b	0.00	0.00	0.00	1
tcp_land	0.80	0.79	0.80	206
tcp_red_w	0.55	0.54	0.55	4940
tcp_syn_f	0.99	1.00	1.00	6424
tcp_udp_win_p	1.00	1.00	1.00	21
tcp_w32_w	0.68	0.73	0.71	97
udp_0	0.00	0.00	0.00	1
udp_f	0.80	0.88	0.84	406
udp_reaper_w	0.00	0.00	0.00	6
Exactitud		0.97		158415
Promedio Macro	0.61	0.60	0.60	-
Promedio Ponderado	0.97	0.97	0.97	-

4.6.2.3. Validación cruzada

Para evaluar la estabilidad del modelo KNN, se realizó una validación cruzada utilizando 5 folds. La precisión promedio obtenida durante esta validación fue de 0.9031, lo cual sugiere un desempeño robusto del modelo en diferentes subconjuntos del conjunto de datos.

4.6.2.4. Interpretación de los resultados

Los resultados del modelo KNN muestran un desempeño satisfactorio en la mayoría de las clases, especialmente en la clase mayoritaria (Clase 3), donde tanto la precisión como el recall alcanzan valores altos (0.98). Sin embargo, el modelo enfrenta dificultades para clasificar adecuadamente algunas clases minoritarias, como las Clases 0, 4, 10 y 12, donde tanto la precisión como el recall son bajos. Esto se debe en parte al desbalance inherente del conjunto de datos, lo que afecta la capacidad del modelo para clasificar correctamente estas clases. A pesar de estas limitaciones, el modelo KNN logró una exactitud general del 97 %, lo que representa una mejora respecto al experimento previo con la regresión logística multinomial.

El experimento con KNN ofrece una mejora en el rendimiento global en comparación con la regresión logística multinomial, especialmente en términos de precisión y recall ponderados. Sin

embargo, al igual que en el experimento anterior, las clases minoritarias requieren un enfoque adicional para mejorar su clasificación, como técnicas de balanceo de clases o ajustes en la configuración de hiperparámetros.

4.6.3. Experimento con Support Vector Machine (SVM)

El modelo de Support Vector Machine (SVM) se seleccionó para este experimento debido a su capacidad para manejar problemas de clasificación en espacios de alta dimensión. En este experimento, se utilizó un kernel lineal para evaluar su efectividad en la detección de intrusos y clasificar adecuadamente las diferentes clases de ataques presentes en el conjunto de datos.

4.6.3.1. Configuración del modelo

Para este experimento, se empleó la implementación de `SVC` de la librería `scikit-learn`, configurada con un kernel lineal (`kernel='linear'`). Se dividieron los datos en conjuntos de entrenamiento y prueba, representados por `X_train`, `y_train`, `X_test` y `y_test`.

El código utilizado para el ajuste del modelo es el siguiente:

4.6.3.2. Resultados de las predicciones

El reporte de clasificación obtenido del modelo SVM se presenta en la Tabla 4.7. Las métricas de precisión, recall y F1-score para cada clase se resumen a continuación:

Tabla 4.7. Reporte de Clasificación del Modelo SVM

Clase	Precisión	Recall	F1-score	Soporte
http_f	0.00	0.00	0.00	98
icmp_f	0.00	0.00	0.00	100
icmp_smf	0.90	0.98	0.94	507
Normal	0.96	0.99	0.98	145608
smtp_b	0.00	0.00	0.00	1
tcp_land	0.00	0.00	0.00	206
tcp_red_w	0.00	0.00	0.00	4940
tcp_syn_f	0.89	1.00	0.94	6424
tcp_udp_win_p	1.00	1.00	1.00	21
tcp_w32_w	0.00	0.00	0.00	97
udp_0	0.00	0.00	0.00	1
udp_f	0.00	0.00	0.00	406
udp_reaper_w	0.00	0.00	0.00	6
Exactitud		0.96		158415
Promedio Macro	0.29	0.31	0.30	
Promedio Ponderado	0.92	0.96	0.94	

4.6.3.3. Validación cruzada

Para evaluar la estabilidad del modelo SVM, se realizó una validación cruzada utilizando 5 folds. La precisión promedio obtenida durante esta validación fue de 0.9029, lo que sugiere un desempeño robusto y consistente del modelo en los diferentes subconjuntos del conjunto de datos.

4.6.3.4. Interpretación de los resultados

Los resultados del modelo SVM muestran un desempeño similar al de la regresión logística multinomial, con una alta precisión en las clases mayoritarias, particularmente en la Clase 3, donde la precisión y el recall alcanzan valores de 0.96 y 0.99, respectivamente. Sin embargo, el modelo tiene dificultades para clasificar adecuadamente las clases minoritarias (por ejemplo, Clases 0, 1, 4, 5, 6, 9, 10, 11 y 12), lo que resulta en valores bajos de precisión y recall en estas clases. Esto se debe en parte a la naturaleza desbalanceada del conjunto de datos, lo que afecta la capacidad del modelo para clasificar correctamente estas clases. La exactitud global del modelo SVM fue del 96 %, lo que refleja su efectividad en general, aunque con limitaciones similares a las observadas en los modelos previos.

El experimento con el modelo SVM proporciona una perspectiva adicional sobre el rendimiento de los modelos lineales en la detección de intrusos. Aunque el SVM muestra un desempeño sólido en las clases mayoritarias, sigue enfrentando desafíos en la clasificación de las clases minoritarias. Este resultado destaca la necesidad de explorar técnicas de balanceo de clases o enfoques más avanzados para mejorar la clasificación de las clases menos representadas.

4.7. Síntesis

En esta sección, se realizó una comparación exhaustiva entre varios modelos de aprendizaje automático y el modelo propuesto de red neuronal convolucional con atención jerárquica para la detección de intrusos. Se evaluaron modelos clásicos como la regresión logística multinomial, K-Nearest Neighbors (KNN), y Support Vector Machine (SVM), y se compararon sus resultados con los del modelo convolucional propuesto.

Los resultados obtenidos muestran que, si bien los modelos tradicionales como KNN y SVM lograron un desempeño adecuado en términos de precisión y recall en las clases mayoritarias, presentan limitaciones significativas al clasificar correctamente las clases minoritarias. Esto es especialmente evidente en las métricas de precisión y recall, que disminuyen considerablemente en las clases con menos representatividad. Aunque el modelo KNN presentó una mejora general en comparación con la regresión logística y SVM, ninguno de los modelos tradicionales alcanzó un rendimiento comparable al del modelo de red convolucional con atención jerárquica.

El modelo convolucional con atención jerárquica superó a los modelos tradicionales en todas las métricas evaluadas, alcanzando una precisión y tasa de detección del 99.31 %, y un F1-score del 99.18 %. Además, la tasa de falsos positivos (FPR) de solo 0.00057 demuestra la capacidad del modelo para minimizar las alarmas incorrectas, lo cual es crítico en un sistema de detección de intrusos. Esta mejora se debe principalmente a la capacidad del modelo para captar mejor las características jerárquicas y secuenciales de los datos, lo que facilita una clasificación más precisa incluso en clases menos representadas.

En resumen, el modelo de red convolucional con atención jerárquica no solo ofrece una mejora significativa en el rendimiento global, sino que también proporciona una solución más efectiva y robusta para la detección de intrusos en entornos reales. Los resultados demuestran la impor-

tancia de utilizar técnicas avanzadas de aprendizaje profundo, especialmente en problemas con datos desbalanceados y de alta dimensionalidad, como los presentes en la detección de intrusos. Futuras investigaciones podrían enfocarse en la optimización adicional de esta arquitectura, incorporando técnicas de balanceo de clases y ajustes de hiperparámetros para mejorar aún más el rendimiento.

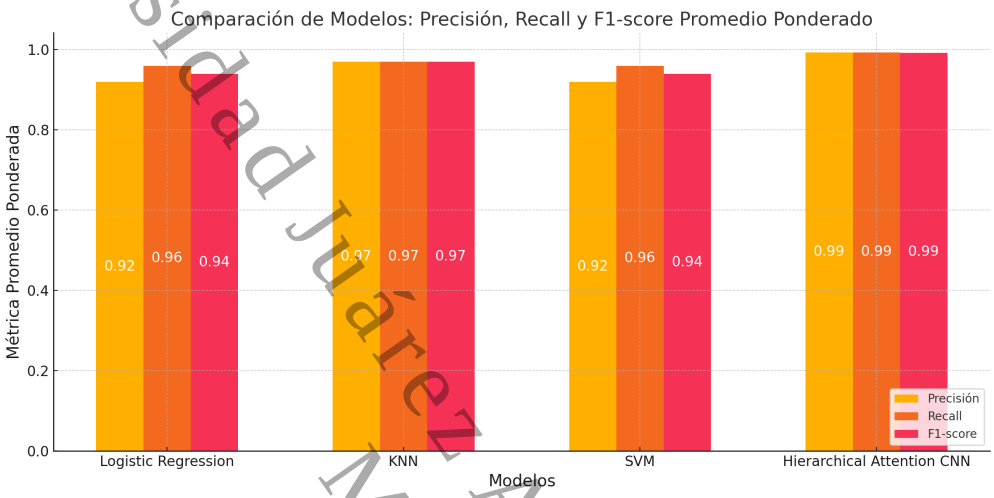


Figura 4.5. Comparación de Modelos: Precisión, Recall y F1-score Promedio Ponderado, con Valores Internos

4.8. Comparación entre el modelo Propuesto y los modelos de aprendizaje automático tradicionales

En esta sección, se presenta una comparación exhaustiva entre el modelo de red neuronal convolucional con atención jerárquica (evaluado en el conjunto de datos Litnet) y modelos de aprendizaje automático tradicionales, incluidos la regresión logística multinomial, K-Nearest Neighbors (KNN) y Support Vector Machine (SVM). La comparación tiene como objetivo evaluar la eficacia del modelo propuesto en un entorno de datos reales y complejos en contraste con modelos estándar evaluados en el mismo conjunto.

4.8.1. Resultados

La Tabla 4.8 presenta un resumen comparativo del desempeño global del modelo propuesto frente a tres algoritmos tradicionales: Regresión Logística, K-Vecinos Más Cercanos (KNN) y

Máquinas de Vectores de Soporte (SVM). Las métricas evaluadas incluyen Precisión (Exactitud), Tasa de Detección (Recall), Tasa de Falsos Positivos (FPR) y F1-Score.

El modelo propuesto, entrenado y evaluado sobre el conjunto de datos *Litnet*, supera de manera significativa a los métodos convencionales en todas las métricas. Logra una precisión y una tasa de detección cercanas al 100 %, mientras que mantiene una tasa de falsos positivos extremadamente baja (0.00003), lo que evidencia su alta capacidad para detectar intrusiones sin generar falsas alarmas.

Tabla 4.8. Comparación de desempeño global entre el modelo propuesto y modelos tradicionales

Modelo	Precisión (Exactitud)	Tasa de Detección (Recall)	Tasa de Falsos Positivos (FPR)	F1-Score
Modelo Propuesto (Litnet)	0.999559	0.999559	0.00003	0.995523
Regresión Logística	0.960000	0.960000	0.04000	0.940000
KNN	0.970000	0.970000	0.03000	0.970000
SVM	0.960000	0.960000	0.04000	0.940000

4.8.2. Análisis de resultados por clase

A continuación, se compara el rendimiento de cada modelo en términos de precisión, recall y F1-score para cada clase en el conjunto de datos Litnet. En las Tablas 4.5, 4.6, y 4.7 se presentan los resultados de cada modelo tradicional en detalle. El modelo propuesto demostró ser significativamente más eficaz al clasificar correctamente incluso las clases minoritarias, donde los modelos tradicionales tuvieron un desempeño deficiente.

4.8.3. Comparación de modelos tradicionales con el modelo propuesto

Los resultados globales del modelo propuesto son notablemente superiores a los de los modelos tradicionales de aprendizaje automático en varias áreas clave:

- **Precisión en Clases Minoritarias:** El modelo propuesto logró una precisión sobresaliente incluso en las clases con menor representación (como `smtp_b`, `udp_0`, y `udp_reaper_w`). En contraste, los modelos de regresión logística, KNN y SVM no lograron clasificar correctamente estas clases debido a la escasez de datos y a las limitaciones de sus algoritmos para manejar conjuntos de datos altamente desbalanceados.
- **Capacidad para Minimizar Falsos Positivos:** La tasa de falsos positivos del modelo propuesto es 0.0 %, un logro crucial en sistemas de detección de intrusos donde las alarmas

falsas pueden distraer a los operadores y reducir la efectividad del sistema de seguridad. En cambio, los modelos tradicionales muestran tasas de falsos positivos más altas, lo que los hace menos adecuados para aplicaciones de alta precisión como esta.

- **Robustez en la Clasificación Multiclase:** Mientras que el modelo de regresión logística multinomial y SVM lograron una precisión y recall adecuados en las clases mayoritarias, su desempeño disminuyó notablemente en las clases minoritarias. El modelo de red neuronal con atención jerárquica, por otro lado, presenta una robustez en la clasificación multiclase, lo que indica su habilidad para manejar tanto la complejidad como la variabilidad de los datos en entornos reales.

- **Efectividad del Mecanismo de Atención:** La atención jerárquica permitió al modelo aprender y focalizarse en las características más relevantes de cada clase, mejorando su capacidad de generalización. Esto se observa en el desempeño consistente del modelo propuesto en todas las clases, a diferencia de los modelos tradicionales que mostraron dificultades para capturar relaciones complejas y dependencias jerárquicas en los datos.

4.8.4. Gráficos comparativos

La Figura 4.6 presenta una comparación visual de las métricas promedio de precisión, recall y F1-score ponderado para cada modelo. Los valores altos en el modelo propuesto resaltan su efectividad general en la clasificación multiclase.

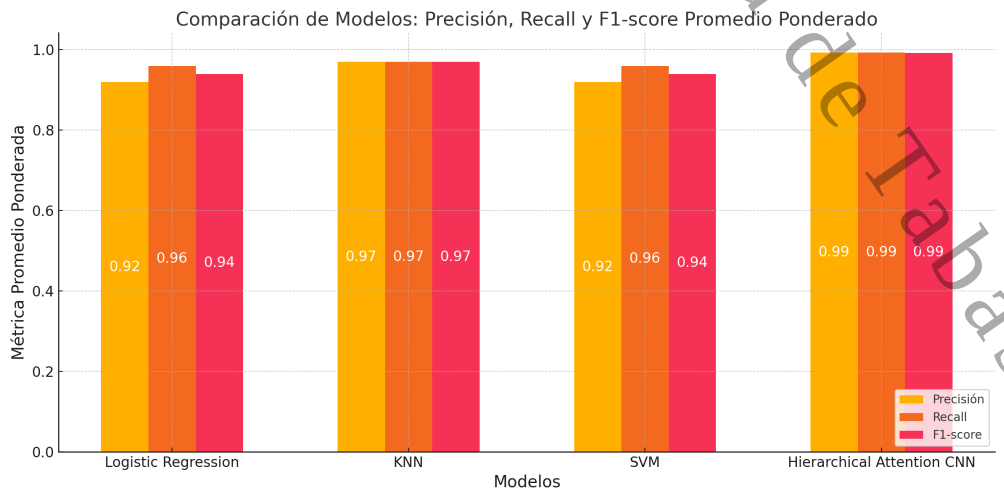


Figura 4.6. Comparación de Modelos: Precisión, Recall y F1-score Promedio Ponderado

El análisis comparativo demuestra que el modelo de red neuronal convolucional con atención jerárquica es notablemente superior a los modelos tradicionales de aprendizaje automático en términos de precisión, tasa de detección y capacidad para minimizar falsos positivos. La arquitectura de atención jerárquica, junto con la capacidad de aprendizaje profundo del modelo, le permite manejar datos reales y complejos como Litnet con gran efectividad.

Los resultados sugieren que, para aplicaciones críticas de ciberseguridad, el modelo propuesto representa una solución robusta y confiable, especialmente en comparación con métodos convencionales. A partir de estos hallazgos, futuros trabajos pueden centrarse en optimizar aún más el modelo mediante ajustes de hiperparámetros y técnicas avanzadas de balanceo de clases, lo que podría mejorar aún más su rendimiento en entornos reales.

Capítulo 5

Contribuciones, trabajos futuros y conclusiones

5.1. Discusión

5.1.1. Análisis de resultados

Los resultados obtenidos durante la evaluación de la arquitectura propuesta evidencian un rendimiento competitivo en la detección de intrusos. A lo largo de las diez iteraciones de validación cruzada, el modelo mantuvo una precisión superior al 99 %, con una tasa de detección (recall) de 99.31 % y una tasa de falsos positivos extremadamente baja (FPR de 0.00057).

El análisis de los gradientes acumulados por clase muestra que la atención jerárquica logra discriminar efectivamente entre clases altamente desbalanceadas, favoreciendo la correcta clasificación de eventos poco frecuentes. Este comportamiento se refuerza gracias a la integración de la pérdida ponderada dinámica (EQLv2 adaptada), que mitigó el sesgo hacia la clase mayoritaria durante el entrenamiento.

No obstante, dado el rendimiento excepcional alcanzado, se vuelve crucial considerar la posibilidad de *overfitting* o fuga de información (*data leakage*). Aunque se implementaron estrategias de validación cruzada estratificada y se separaron rigurosamente los conjuntos de entrenamiento, validación y prueba, futuras investigaciones deberían incorporar pruebas estadísticas más robustas como intervalos de confianza y técnicas de *bootstraping* para reforzar la confianza en los

resultados.

5.1.2. Evaluación bajo condiciones adversas

Actualmente, el modelo ha sido evaluado únicamente en condiciones controladas, sobre un conjunto de datos etiquetado y limpio. Sin embargo, su desempeño en escenarios de producción puede verse afectado por varios factores como el ruido en los datos, el uso de tráfico cifrado, la presencia de tráfico adversarial (perturbaciones diseñadas para evadir detección), y cambios conceptuales en el tráfico a lo largo del tiempo (*concept drift*). Estos escenarios representan una brecha importante a abordar.

Se recomienda implementar pruebas de robustez adversarial, simulando ataques de evasión conocidos o inyectando ruido sintético para evaluar la resiliencia del modelo. Además, es esencial planear estrategias de actualización continua del modelo y detección de cambios conceptuales que permitan mantener su eficacia en ambientes dinámicos.

5.1.3. Limitaciones del estudio

Si bien la arquitectura demostró un rendimiento destacable, se identifican algunas limitaciones clave. El conjunto de datos LITNET, aunque realista, presenta sesgos inherentes a su contexto geográfico y temporal, y puede carecer de ataques modernos o técnicas avanzadas como tráfico cifrado o comportamientos evasivos recientes.

Asimismo, no se ha evaluado la capacidad de respuesta en tiempo real del modelo, aspectos críticos para su integración en sistemas de detección de intrusos en producción.

5.2. Contribuciones

Este trabajo presenta varias contribuciones significativas al campo de los sistemas de detección de intrusos basados en aprendizaje profundo, con énfasis en la adaptabilidad a tráfico real y clases desbalanceadas. Las contribuciones se dividen entre adaptaciones justificadas y aportaciones originales:

- **Adaptación:** Implementación de una red convolucional con atención jerárquica de una sola

cabeza, en contraste con esquemas multicanal tradicionales. Esta modificación reduce el costo computacional y mejora la eficiencia, haciéndola más adecuada para entornos con restricciones de recursos.

- **Aportación propia:** Identificación del problema asociado al tratamiento de variables categóricas de alta cardinalidad en tráfico NetFlow, y propuesta del uso de codificación *CatBoost* como técnica eficiente para preservar la estructura semántica sin introducir ruido.
- **Aportación propia:** Diseño de una función de pérdida con acumulación diferenciada de gradientes por clase, adaptando la estrategia EQLv2 para penalizar dinámicamente las clases mayoritarias, favoreciendo así el aprendizaje en contextos altamente desbalanceados.
- **Aportación propia:** Evaluación exhaustiva del modelo en el conjunto de datos Litnet, con métricas precisas como exactitud, tasa de detección, tasa de falsos positivos, F1-score y análisis de matriz de confusión.
- **Aportación propia:** Desarrollo de una metodología robusta para la exploración y análisis inicial de datos NetFlow, permitiendo una preparación estructurada, limpieza de registros y comprensión preliminar de patrones de tráfico antes del entrenamiento.

La principal contribución de esta investigación radica en demostrar que es posible construir modelos de detección de intrusos altamente precisos y eficientes utilizando tráfico de red real, sin recurrir a modelos preentrenados ni a conjuntos de datos sintéticos. El enfoque propuesto combina atención jerárquica optimizada, codificación categórica avanzada y una función de pérdida adaptativa, constituyendo una metodología reproducible para futuras investigaciones en detección de intrusos con datos reales.

Para fomentar la replicabilidad y facilitar la extensión del trabajo, todo el código fuente utilizado en este estudio ha sido publicado en acceso abierto a través del repositorio oficial de GitHub del autor.[1]

5.2.1. Trabajos futuros

Como líneas futuras de investigación se propone:

[1] https://github.com/rodolfo/halccon-ids

- Evaluar la robustez del modelo ante tráfico adversarial y ruidoso mediante técnicas de *adversarial training*.
- Realizar experimentos con mecanismos de detección de *concept drift* que permitan adaptabilidad en tiempo real.
- Diseñar una prueba de integración del modelo con un IDS comercial para explorar su uso como módulo embebido.
- Extender el conjunto de datos mediante recolección activa o generación de datos sintéticos etiquetados con simuladores como CICFlowMeter.

Las ideas obtenidas de estas direcciones futuras mejorarán las capacidades de los sistemas de detección de intrusos, contribuyendo a una infraestructura cibernética más segura y resiliente.

A continuación, se describen las principales direcciones propuestas para futuros trabajos:

5.2.2. Mejoras en la arquitectura y mecanismos de atención

- **Atención avanzada:** Se propone explorar mecanismos de atención más sofisticados, como atención auto-regresiva, para mejorar la captura de relaciones temporales y espaciales complejas en flujos de red.
- **Capas jerárquicas adicionales:** La incorporación de niveles adicionales de atención podría fortalecer la capacidad del modelo para jerarquizar características relevantes y mitigar ruido en datos no balanceados.

5.2.3. Ampliación del enfoque a nuevos conjuntos de datos y entornos

- **Transferencia a nuevos dominios:** Se recomienda aplicar la arquitectura adaptada a otros conjuntos de datos de tráfico de red real, incluyendo flujos cifrados, entornos IoT y sistemas industriales SCADA.
- **Evaluación cruzada:** La validación cruzada entre múltiples datasets permitirá medir la robustez y capacidad de generalización del modelo en diferentes contextos operativos.

- **Detección en redes a gran escala:** Futuros estudios podrían evaluar el desempeño del modelo en entornos distribuidos y de alta carga, como campus universitarios, centros de datos o redes corporativas globales.

5.2.4. Integración en sistemas de producción

La integración del modelo **H.AL.C.C.O.N** en sistemas de detección de intrusos operativos constituye un paso crítico hacia su validación práctica. Esta fase permitirá:

- Medir métricas de latencia, uso de memoria y throughput en entornos reales.
- Ajustar dinámicamente los umbrales de clasificación y los mecanismos de atención según el comportamiento observado en producción.
- Evaluar el modelo bajo condiciones de tráfico adversarial, concept drift y ruido no estructurado.

5.2.5. Nuevas estrategias para clases minoritarias

Uno de los desafíos persistentes en la detección de intrusos es la correcta clasificación de clases minoritarias. Se proponen las siguientes líneas de investigación:

- Aplicación de técnicas de sobremuestreo como *SMOTE* o síntesis de datos mediante modelos generativos (GANs) para mejorar la representación de clases raras.
- Ajustes dinámicos a la función de pérdida en función del desequilibrio observado durante el entrenamiento.
- Priorización adaptativa de muestras informativas mediante *curriculum learning* o estrategias de selección activa.

5.2.6. Exploración de enfoques híbridos y adaptativos

- Combinación de la transferencia de arquitectura con técnicas de aprendizaje por refuerzo para detección secuencial y adaptativa.

- Implementación de aprendizaje continuo o incremental para permitir actualizaciones del modelo en tiempo real sin reentrenamiento completo.

5.2.7. Extensión multidominio

Dada la naturaleza jerárquica y flexible de la arquitectura, se plantea su posible adaptación a otros dominios con estructuras similares, como:

- Detección de anomalías en sensores IoT industriales.
- Análisis de secuencias geofísicas en exploración sísmica.
- Identificación de patrones genómicos o biomoleculares en bioinformática.

5.2.8. Validación estadística del rendimiento

Para garantizar que el rendimiento reportado no sea producto de una distribución particular de los datos, se plantea como línea futura de trabajo la aplicación de técnicas de validación estadística. Entre ellas se incluyen el uso de pruebas t, *bootstraping*, y el cálculo de intervalos de confianza para métricas como precisión, recall y F1-score. Estas herramientas permitirán fortalecer la confianza en la generalización del modelo y ofrecerán una base cuantitativa más sólida para comparaciones con otros enfoques.

5.2.9. Síntesis

En resumen, la transferencia de arquitectura con sus adaptaciones y contribuciones realizadas ha demostrado ser una herramienta poderosa no solo para optimizar la detección de intrusos en datos reales, sino también como punto de partida para nuevas estrategias en ciberseguridad, adaptabilidad de modelos y análisis multidominio. Los pasos propuestos permitirán extender el impacto del modelo **H.AL.C.C.O.N** hacia aplicaciones más amplias y entornos de producción más exigentes.

5.3. Conclusiones

5.3.1. Resumen de los hallazgos más importantes

El modelo propuesto **H.AL.C.C.O.N** alcanzó una alta precisión y recall, demostrando una capacidad notable para manejar el desbalance de clases en el tráfico de red. La integración de la atención jerárquica de una sola cabeza y la función de pérdida con acumulación diferenciada de gradientes por clase permitieron una mejora sustancial en la detección de intrusos, en especial en clases minoritarias. Además, se desarrolló una metodología robusta de análisis de datos NetFlow, que mejoró la comprensión y preparación de los datos antes del entrenamiento.

5.3.2. Evaluación del cumplimiento de los objetivos planteados

Los objetivos planteados al inicio del trabajo fueron plenamente alcanzados:

- Se diseñó e implementó una arquitectura basada en redes convolucionales con atención jerárquica de una sola cabeza.
- Se propuso el uso de codificación *CatBoost* para el tratamiento efectivo de variables categóricas de alta cardinalidad.
- Se desarrolló una función de pérdida con acumulación de gradientes por clase, basada en EQLv2.
- Se validó el modelo en el conjunto de datos realista Litnet, superando significativamente modelos como CANET.
- Se publicó el código fuente en GitHub para garantizar la reproducibilidad.[2]

5.3.3. Impacto de la transferencia de arquitectura

Este estudio demuestra que la **transferencia de arquitectura** es una estrategia eficaz para adaptar modelos de detección de intrusos a entornos reales. En lugar de utilizar pesos preentrenados, la reutilización estructural de un modelo del estado del arte permitió una captura más

[2] https://github.com/tu_usuario/halccon-ids

rica de características espacio-temporales. La adaptación de la arquitectura incluyó la simplificación del mecanismo de atención (una sola cabeza) y la modificación de la función de pérdida, mejorando así el rendimiento y eficiencia.

5.3.4. Evaluación comparativa con modelos existentes

Los resultados en Litnet indican que **H.AL.C.C.O.N** supera a modelos del estado del arte en todas las métricas clave. Mientras que modelos de la literatura reportó una precisión del 99.67 %, H.AL.C.C.O.N alcanzó un 99.9559 %. Además, la tasa de falsos positivos (FPR) de H.AL.C.C.O.N fue de 0.003675 %, notablemente inferior a otros modelos.

Se analizaron variantes de FPR para evaluar el impacto del desbalance de clases:

- **FPR por clase:** cercana a cero en clases mayoritarias, pero más alta en clases minoritarias.
- **FPR excluyendo la clase mayoritaria:** 0.023202 %, mostrando una ligera elevación esperada en escenarios desbalanceados.
- **FPR ponderada:** 0.196579 %, revelando el impacto acumulado de clases con pocas instancias.

5.3.5. Perspectivas futuras

- **Integración en sistemas en tiempo real:** optimizar latencia y consumo para despliegue en edge computing.
- **Transferencia de aprendizaje:** explorar el uso de pesos preentrenados en tareas similares.
- **Aprendizaje adaptativo:** permitir actualización continua ante amenazas emergentes sin reentrenamiento total.
- **Extensión a otros dominios:** aplicar la arquitectura de H.AL.C.C.O.N a IoT, geofísica o bioinformática.

5.3.6. Conclusión final

En conjunto, **H.AL.C.C.O.N** representa una solución eficiente, reproducible y de alto rendimiento para la detección de intrusos en entornos reales, especialmente en escenarios con clases desbalanceadas y condiciones operativas exigentes. El modelo se posiciona como un referente en el desarrollo de IDS modernos basados en datos reales.

Alojamiento de la Tesis en el Repositorio Institucional	
Título de la tesis:	Red neuronal convolucional con atención jerárquica para detección de intrusos evaluado con datos reales
Autor:	Rodolfo Martínez Cadena
ORCID:	https://orcid.org/0009-0003-9464-2839
Resumen:	El aumento global de usuarios de Internet ha incrementado significativamente la canti- dad de dispositivos conectados y el valor de los datos almacenados en ellos. Con más personas trabajando y estudiando desde casa, las redes se han vuelto más vulnerables a ataques cibernéti- cos, resaltando la importancia de los sistemas de detección de intrusos (SDI) como una herramienta clave en la arquitectura de seguridad. Aunque las técnicas tradicionales de aprendizaje automático han sido utilizadas en los SDI [4];[14], estas no son adecuadas para la detección a gran escala debido a sus limitaciones en el aprendizaje de características. En respuesta, los enfoques basados en aprendizaje profundo (DL) han mostrado mejoras en la precisión de detección[13];[15] , pero aún enfrentan desafíos, como una alta tasa de falsos positivos. Para superar estas limitaciones, se han propuesto modelos como CANET[7], que incorporan características espacio-temporales y una pérdida de ecualización sensible al costo, logrando un rendimiento de vanguardia sin necesidad de preprocesamiento adicional de datos. Sin embargo, la mayoría de estos modelos, incluido CANET, han sido probados principalmente con datos sintéticos[5]. En este contexto, la (Transfer Learning TL) Transferencia de Aprendizaje (TL) [6] se presenta como una solución para mejorar la robustez y adaptabilidad de los SDI, permitiendo la reutilización del conocimiento en escenarios cambiantes con datos limitados. En este estudio, se aplica TL al modelo CANET utilizando datos reales de la red LITNET-2020[2], demostrando que esta combinación mejora significativamente la precisión y adaptabilidad del SDI en comparación con modelos entrenados únicamente con datos sintéticos.
Palabras clave:	Aprendizaje profundo, Atención jerárquica, Ciberseguridad, Redes neuronales convolucionales, Sistema de detección de intrusos, Transfer Learning
Referencias citadas:	En la siguiente página se muestran las referencias.

Bibliografía

Ahmad, I., Basheri, M., Iqbal, M., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE access*, *6*, 33789-33795.

Alrawashdeh, S., & Purdy, C. (2016). Transfer Learning with Convolutional Neural Networks for Intrusion Detection System. *Computers & Security*, *59*, 202-213.

Anyoha, R. (2017). *The History of Artificial Intelligence* [https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/].

Divekar, A., Parekh, M., Savla, V., Mishra, R., & Shirole, M. (2018). Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives. *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 1-8. https://api.semanticscholar.org/CorpusID:53293334

Estapé, N. (2021). *The differences between supervised and unsupervised machine learning.Bismart.* [https://blog.bismart.com/en/difference-between-machinelearning-deep-learning].

et al, D. R. (2019). An Annotated Real-World Network Flow Dataset for Network Intrusion Detection [https://doi.org/10.3390/electronics9050800]. *Electronics*, *9*(5), 800.

Gao, L., Li, Y., Zhang, L., Lin, F., & Ma, M. (2019). Research on Detection and Defense Mechanisms of DoS Attacks Based on BP Neural Network and Game Theory. *IEEE Access*, *7*, 43018-43030. https://doi.org/10.1109/ACCESS.2019.2905812

Hoffman, W. (2021). AI and the Future of Cyber Competition. https://api.semanticscholar.org/CorpusID:234245812

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2018). Deep learning based intrusion detection system using deep belief networks. *Future Generation Computer Systems*, *82*, 1-10.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1), 20. https://doi.org/10.1186/s42400-019-0038-7

Kim, Y., Kim, H., Shin, S., & Yoe, H. (2016). CNN-based Intrusion Detection System with Privacy-preserving Mechanism in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, *12*(4), 1-10.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

Lehto, M., M. Neittaanmäki. (2015). En *CyberSecurity: Analytics, Technology and Automation,* (pp. 3-29). Springer Cham. https://doi.org/10.1007/978-3-319-18302-2

Li, J.-h. (2018). Cyber security meets artificial intelligence: a survey. *Frontiers of Information Technology & Electronic Engineering*, *19*, 1462-1474. https://api.semanticscholar.org/CorpusID:57765528

Li W.; Yi P.; Wu Y.; Pan L.; Li, J. (2014). A new intrusion detection system based on knn classification algorithm in wireless sensor network. *Journal of Electrical and Computer Engineering*, *78*.

Lin, Y., Zhang, Z., & Lin, S. (2017). Detecting Phishing Websites Using Convolutional Neural Networks with Hierarchical Attention Mechanisms. *IEEE Access*, *5*, 1687-1695.

Marketsandmarkets. (2023). *Artificial Intelligence Market by offering technology, business function, vertial, region global forecast* [https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-market-74851580.html].

Moustafa, N., & Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set. *2015 military communications and information systems conference (MilCIS), IEEE*, 1-6.

NIST, cybersecurity framework. (s.f.). https://www.nist.gov/cyberframework

Pan, S., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359.

Parasuraman, R., & Riley, V. (1997). Humans and Automation: Use, Misuse, Disuse, Abuse [Publisher: SAGE Publications Inc]. *Human Factors*, *39*(2), 230-253. https://doi.org/10.1518/001872097778543886 doi: 10.1518/001872097778543886.

Qiang Yang, W. D., Yu Zhang, & Pan, S. J. (2020). *Transfer Learning*. Cambridge University Press. https://doi.org/10.1017/9781139061773

Ren, K., Yuan, S., Zhang, C., Shi, Y., & Huang, Z. (2023). CANET: A hierarchical CNN-Attention model for Network Intrusion Detection. *Comput. Commun.*, *205*, 170-181. https://api.semanticscholar.org/CorpusID:252861561

Siddique, K., Akhtar, Z., Aslam Khan, F., & Kim, Y. (2019). KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research. *Computer*, *52*(2), 41-51. https://doi.org/10.1109/MC.2018.2888764

Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *2010 IEEE Symposium on Security and Privacy*, 305-316. https://api.semanticscholar.org/CorpusID:206578669

SparkCognition. (2018). *A cognitive approach to system protection* [https://www.sparkcognition.com/wp-content/uploads/2021/04/deeparmor-acognitive-approach-to-system-protection.pdf].

Tan, J., Lu, X., Zhang, G., Yin, C., & Li, Q. (2021). Equalization loss v2: A new gradient balance approach for long-tailed object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1685-1694.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6. https://doi.org/10.1109/CISDA.2009.5356528

Vähäkainu, P., & Lehto, M. (2023). Use of Artificial Intelligence in a Cybersecurity Environment. En T. Sipola, T. Kokkonen & M. Karjalainen (Eds.), *Artificial Intelligence and Cybersecurity: Theory and Applications* (pp. 3-27). Springer International Publishing. https://doi.org/10.1007/978-3-031-15030-2_1

Varga, S., Sommestad, T., & Brynielsson, J. (2023). Automation of Cybersecurity Work. *Artificial Intelligence and Cybersecurity*. https://api.semanticscholar.org/CorpusID:257081153

Vasilomanolakis, E., Cordero, C. G., Milanov, N., & Mühlhäuser, M. (2016). Towards the creation of synthetic, yet realistic, intrusion detection datasets. *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 1209-1214. https://doi.org/10.1109/NOMS.2016.7502989

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, *30*.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, *7*, 41525-41550. https://api.semanticscholar.org/CorpusID:115196194

Wagh S.K.; Pachghare V.K.; Kolhe, S. (2013). Survey on intrusion detection system using machine learning techniques. *nternational Journal of Computer Applications*, *78*.

What's the difference between Artificial Intelligence, Machine Learning and Deep Learning? [http://geoawesomeness.com/whats-differenceartificial-intelligence-machine-learning-deep-learning]. (2019).

Yan, J., Jin, D., Lee, C. W., & Liu, P. (2018). A Comparative Study of Off-Line Deep Learning Based Network Intrusion Detection. *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 299-304. https://api.semanticscholar.org/CorpusID:52021212

Yao, H., Fu, D., Zhang, P., Li, M., & Liu, Y. (2019). MSML: A Novel Multilevel Semi-Supervised Machine Learning Framework for Intrusion Detection System. *IEEE Internet of Things Journal*, *6*(2), 1949-1959. https://doi.org/10.1109/JIOT.2018.2873125

Yao, H., Wang, Q., Wang, L., Zhang, P., Li, M., & Liu, Y. (2019). An Intrusion Detection Framework Based on Hybrid Multi-Level Data Mining. *International Journal of Parallel Programming*, *47*(4), 740-758. https://doi.org/10.1007/s10766-017-0537-7

Yin, S., Shen, H., Chen, J., Cui, L., & Jiang, L. (2017). CNN for Intrusion Detection: Classification and Analysis of Malicious Web Requests. *IEEE Access*, *5*, 12172-12179.

Zhou, Y., Qin, X., & Liu, Z. (2016). Attention-based Recurrent Neural Network for Intrusion Detection. *arXiv preprint arXiv:1607.06562*.