FACULTAD DE ARQUITECTURA E INGENIERÍAS

Trabajo de fin de carrera titulado:

"ESTUDIO COMPARATIVO ENTRE MODELOS DE APRENDIZAJE PROFUNDO, DESARROLLADOS A PARTIR DE REDES NEURONALES RECURRENTES Y REDES NEURONALES CONVOLUCIONALES, PARA LA DETECCIÓN DE INTRUSOS DE RED"

Realizado por:

David Alejandro Jarrín Rodríguez

Director del proyecto:

Ing. Diego Fernando Riofrío Luzcando, PhD.

Como requisito para la obtención del título de:

MAGISTER EN TECNOLOGÍAS DE LA INFORMACIÓN CON MENCIÓN EN SEGURIDAD DE REDES Y COMUNICACIÓN

Quito, 06 de septiembre 2019

DECLARACIÓN JURAMENTADA

Yo, **DAVID ALEJANDRO JARRÍN RODRÍGUEZ**, con cédula de identidad **172258253-1**, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido previamente presentado para ningún grado a calificación profesional; y, que ha consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

David Alejandro Jarrín Rodríguez

C.C: 1722582531

DECLARATORIA

El presente trabajo de investigación titulado:

"ESTUDIO COMPARATIVO ENTRE MODELOS DE APRENDIZAJE PROFUNDO, DESARROLLADOS A PARTIR DE REDES NEURONALES RECURRENTES Y REDES NEURONALES CONVOLUCIONALES, PARA LA DETECCIÓN DE INTRUSOS DE RED"

Realizado por:

DAVID ALEJANDRO JARRÍN RODRÍGUEZ

Como requisito para la Obtención del Título de:

MAGÍSTER EN TECNOLOGÍAS DE LA INFORMACIÓN CON MENCIÓN EN SEGURIDAD DE REDES Y COMUNICACIÓN

Ha sido dirigido por el profesor

ING. DIEGO FERNANDO RIOFRIO LUZCANDO, PhD.

Quien considera que constituye un trabajo original de su autor

Ing. Diego Fernando Riofrio Luzcando, PhD.

DIRECTOR

PROFESORES INFORMANTES

Después de revisar el trabajo presentado, lo ha calificado como apto para su defensa oral ante el tribunal examinador.

Ing. Verónica Rodríguez, MBA.

Ing. Fabián Hurtado, MGS.

Quito, 06 de septiembre de 2019

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

David Alejandro Jarrín Rodríguez

C.C: 1722582531

DEDICATORIA

Dedico este trabajo a mis padres, Alfonso y Beatriz, a hermana, Anita, quienes permanentemente me apoyaron con su espíritu alentador y consejos, contribuyendo incondicionalmente a lograr las metas y objetivos propuestos en toda mi educación, tanto académica, como en la vida. A mi novia Erika, por su amor, apoyo incondicional, comprensión y ejemplo, que me permitieron culminar esta nueva etapa en mi vida.

AGRADECIMIENTO

El apoyo brindado por personas valiosas que, con sus palabras de aliento hacen posible que las metas y objetivos propuestos, sean más fáciles de alcanzar, es por eso que agradezco a mi familia, a mi novia, a mi director Diego Riofrío, y demás maestros, compañeros y amigos que, de una u otra manera, me supieron brindar el apoyo y el incentivo para culminar con éxito esta etapa de mi vida.

RESUMEN

En las últimas décadas, se observa la enorme expansión de la Internet. Debido a esto salen a la luz problemas relacionados con la gestión de la seguridad, que afectan sistemas conectados a la red global. A causa de estos problemas de seguridad, se vuelve primordial incorporar controles de acceso y herramientas de defensa en los sistemas informáticos, las cuales permitan prevenir, evadir y detectar ataques informáticos; como es el caso de los sistemas de detección de intrusos (IDS). Es por esto, que este trabajo tiene el objetivo de comparar dos modelos de aprendizaje profundo utilizados para la identificación de trafico benigno o maligno en una red, modelos que fueron desarrollados mediante Redes Neuronales Recurrentes (RNN) y Redes Neuronales Convolucionales (CNN). Para este fin se utilizó el conjunto de datos CICIDS2017, el cual permitió entrenar ambos modelos de predicción. Estos fueron validados mediante técnicas de validación cruzada, obteniendo una precisión de sobre el 97%. El punto disruptivo para la elección de uno de los dos modelos es el tiempo de entrenamiento de cada red, ya que la CNN mostró ser más rápida que la RNN.

Palabras Clave: Aprendizaje profundo, Red Neuronal Convolucional, Red Neuronal Recurrente, Sistema de Detección de Intrusos, CICIDS2017.

ABSTRACT

In the last decades the expansion of the Internet has shown problems related to security management, affecting systems of the global network. It is because of these security problems that are necessary to incorporate, access control and defense tools in the information systems. These tools can help to prevent, evade and detect computer attacks, as the intrusion detection system (IDS). This investigation has the objective of compare two deep learning models used for the identification of benign or malign traffic in a network, models that were developed through Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). For this purpose the dataset CICIDS2017 was used, helping to train the prediction models. Both models were probed through cross validation techniques, getting accuracy over 97%. The training time of the network is the way for select the best model, considering that the CNN showed to be faster than RNN.

Keywords: Deep Learning, Convolutional Neural Network, Recurrent Neural Network, Intrusion Detection System, CICIDS2017.

ÍNDICE GENERAL DE CONTENIDOS

CAPITULO I 14
INTRODUCCIÓN 14
1.1. El problema de la investigación 14
1.1.1. Planteamiento del problema 15
1.1.2. Formulación del problema 17
1.2. Objetivo general 17
1.2.1. Objetivos específicos 17
1.3. Justificación 18
1.4. Marco Teórico 19
1.4.1. Seguridad Informática 19
1.4.2. Amenazas Informáticas 20
1.4.3. Detección de Ataques Intrusos 22
1.4.4. Técnicas para la Detección de Intrusos 23
1.4.5. Aprendizaje de máquina 24
1.4.5.1. Redes Neuronales Artificiales 26
1.4.5.2. Redes Neuronales Recurrentes (Recurrent Neural Networks - RNN) ... 28
1.4.5.2.1. Long Short-Term Memory (LSTM) 30
1.4.5.3. Redes Neuronales Convolucionales (Convolutional Neural Networks - CNN) 31
1.4.6. DEEP LEARNING 34
CAPITULO II 36
ESTADO DEL ARTE 36
2.1. Aprendizaje de máquina para detección de intrusiones 36
2.1.1. Aprendizaje de máquina para detección de malware 36
2.1.2. Aprendizaje de máquina para detección de red 39
2.1.3. Aprendizaje de máquina para detección de intrusos red utilizando el conjunto de datos de referencia KDD99 y NSL-KDD 45
2.1.4. Aprendizaje de máquina para detección de red utilizando el conjunto de datos de referencia CICIDS2017 66

2.2. Comparativa de redes neuronales para analítica de datos 70
2.2.1. Comparativa entre CNN y RNN 74
2.3. Discusión 76
CAPITULO III 77
SOLUCIÓN ADOPTADA 77
3.1. Introducción 77
3.2. Descripción de la arquitectura 77
3.2.1. Análisis de solución 77
3.2.2. Diseño de solución 78
3.2.2.1. Fuente de Información 79
3.2.2.2. Preparación de los datos 84
3.2.2.3. Aprendizaje Profundo 90
3.2.2.4. Validación de modelos 97
CAPITULO IV 98
4.1. Método 98
4.2. Resultados 100
4.3. Discusión 105
CAPITULO V 110
CONCLUSIONES Y TRABAJOS FUTUROS 110
5.1. Conclusiones 110
5.2. Trabajos Futuros 111
REFERENCIAS BIBLIOGRÁFICAS 113

Índice General de Tablas

Tabla 1. Lista de acrónimos 40
Tabla 2. Investigaciones aprendizaje de máquina (Single classifiers) en sistemas de detección de intrusos. 43
Tabla 3. Investigaciones aprendizaje de máquina (Hybrid classifiers) en sistemas de detección de intrusos. 44
Tabla 4. Investigaciones aprendizaje de máquina (Ensemble classifiers) en sistemas de detección de intrusos. 45
Tabla 5. Investigaciones aprendizaje de máquina (Single classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos KDD-99. 49
Tabla 6. Investigaciones aprendizaje de máquina (Hybrid classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos KDD-99. 60
Tabla 7. Investigaciones aprendizaje de máquina (Ensemble classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos KDD-99. 66
Tabla 8. Investigaciones aprendizaje de máquina (Single classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos CICIDS2017. 67
Tabla 9. Investigaciones aprendizaje de máquina (Hybrid classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos CICIDS2017. 68
Tabla 10. Investigaciones aprendizaje de máquina (Ensemble classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos CICIDS2017. 69
Tabla 11. Sistemas operativos y direcciones IP de red víctima y atacante (Sharafaldin et al., 2018) 81
Tabla 12. Etiquetas de ataque diario CICIDS2017 (Sharafaldin et al., 2018) 82
Tabla 13. Arquitecturas CNN. 94
Tabla 14. Arquitecturas RNN 97
Tabla 15. División de datos para entrenamiento y pruebas 99
Tabla 16. Resumen resultados de precisión, pérdida y tiempo de entrenamiento 105
Tabla 17. Tabla resumen del modelo CNN 106
Tabla 18. Tabla resumen del modelo RNN 107

Índice General de Figuras

Figura 1. Esquema de modelo Neurona (Serrano et al., 2009) 27
Figura 2. Izquierda: Red Parcialmente Recurrente Derecha: Red Completamente recurrente (Gers, 2001) 29
Figura 3. Bloque de memoria LSTM (S. Liu et al., 2018) 30
Figura 4. Obtención de Matriz de Activación (Durán, 2017) 32
Figura 5. Diferencia entre las capas convolucionales 1D y 2D (Ackermann, 2018a) 32
Figura 6. Aplicación de diferentes tipos de Pooling (Durán, 2017) 34
Figura 7. Comparación entre machine learning y deep learning (Carreras, 2018) 35
Figura 8. Arquitectura de la solución adoptada. 78
Figura 9. Diagrama de red CICIDS2017 (Sharafaldin et al., 2018). 81
Figura 10. Arquitectura CNN (Zaragoza, 2018) 90
Figura 11. Arquitectura RNN (Khosla, Ramesh, Sharma, & Nyakotey, 2018) 94
Figura 12. Diagrama de flujo de entrenamiento RNN y CNN 98
Figura 13. Exactitud por cada entrenamiento CNN 101
Figura 14. Perdida por cada entrenamiento CNN 102
Figura 15. Tiempo por cada entrenamiento CNN 102
Figura 16. Exactitud por cada entrenamiento RNN 103
Figura 17. Perdida por cada entrenamiento RNN 104
Figura 18. Tiempo por cada entrenamiento RNN 104
Figura 19. Uso de memoria entrenamientos CNN 108
Figura 20. Uso de memoria entrenamientos RNN 108

CAPÍTULO I

INTRODUCCIÓN

1.1. El problema de la investigación

En las últimas décadas es fácil observar el enorme crecimiento de la Internet, gracias a los avances tecnológicos en el área de las telecomunicaciones y las redes de datos. El crecimiento es tal, que se estima que para el 2020 existirán cerca de 50 mil millones de dispositivos conectados a nivel mundial (Evans, 2011).

Además, Internet se está expandiendo a lugares que hasta ahora eran inalcanzables. Las personas cada vez más mueven aspectos de sus vidas hacia la Internet, confiando que sus datos estarán protegidos contra cualquier violación. Pero al mismo tiempo, el número de casos reportados de ataques informáticos aumentan, y se desconoce el alcance total de los mismos (Internet Society, 2016).

El principal objetivo de los atacantes son organizaciones privadas y gubernamentales, buscando recopilar información que posteriormente permita realizar suplantación de identidad. Los intentos de acceso en su gran mayoría son provenientes de fuentes externas, las cuales buscan explotar vulnerabilidades de seguridad ya conocidas. (Internet Society, 2016).

Moar (2018) estimó que para el año 2019 el costo de los ataques informáticos en los cuales existió violación de datos ascenderá a 2.1 billones de dólares, lo que cuadriplica en casi cuatro veces al costo estimado en el año 2015 que fue de 5 mil millones de dólares.

1.1.1. Planteamiento del problema

Como se mencionó anteriormente, los altos costos que representan las brechas de seguridad, han convertido a la seguridad informática en un tema primordial para las empresas, ya que brinda un conjunto de herramientas, políticas, métodos de gestión, acciones, y prácticas idóneas, para proteger los activos de la organización y sus usuarios en el entorno tecnológico. Pero no es una tarea fácil, debido a la complejidad en aumento de las amenazas y distintos tipos ataques. Las brechas y violaciones de seguridad en una organización afectan gravemente a la reputación de la misma, la confianza de sus clientes y la confidencialidad de su información (Capó & García, 2015).

El crecimiento de Internet antes mencionado, ha sacado a la luz numerosos problemas respecto a la gestión de la seguridad en los sistemas conectados a la misma. Así también el incremento de usuarios con acceso a la red global, ha desencadenado la necesidad de incorporar controles de acceso y herramientas de defensa en los sistemas, que eviten el ingreso de usuarios no autorizados. Estos internautas no autorizados o también llamados intrusos, son atacantes que mediante diferentes métodos y técnicas logran infiltrar o interrumpir el tráfico de una red o un sistema, con un fin específico (Lee, Amaresh, Green, & Engels, 2018).

La compleja situación de las redes informáticas, ha traído consigo un gran avance en desarrollo de investigaciones, cuyo principal objetivo es la creación de mecanismos de seguridad que permitan prevenir, evadir y detectar ataques informáticos (Kruegel, Valeur, & Vigna, 2005).

Una de las principales herramientas para detección de intrusos de red son los *NIDS (Network Intrusion Detection System*), estos sistemas permiten detectar los cambios en el estado de una red, basando su funcionamiento en el monitoreo y análisis del tráfico que circula por la misma. La detección de anomalías presenta ventajas en la identificación de ataques desconocidos a través de la especificación del comportamiento normal de la red. Estos sistemas de detección a través de los años,

han venido adoptando en su funcionamiento diversas técnicas de aprendizaje de máquina, destacándose entre ellas, las redes neuronales artificiales (*ANN*), máquinas de soporte vectorial (*SVM*), Arboles de decisión, Aprendizaje profundo (*Deep Learning*), entre otras (Lorenzo et al., 2009).

De manera general, para la creación de modelos de aprendizaje profundo se utilizan varias capas de redes neuronales jerárquicas que usan distintas técnicas de aprendizaje no supervisado (Lee et al., 2018).

Entre los modelos de aprendizaje profundo se pueden destacar a las redes neuronales convolucionales y las redes neuronales recurrentes, como los modelos más utilizados en sistemas de detección de intrusos (Hindy et al., 2018).

Como se vio anteriormente, existe un alto riesgo de ataque a los sistemas informáticos que se encuentran interconectados a través de Internet. Esto ha generado la necesidad de desarrollar diversos mecanismos de seguridad que permitan prevenir, evadir y detectar estas irrupciones. Una de las principales herramientas para este fin, son los Sistemas de Detección de Intrusos de red, los cuales como fue mencionado, han venido adaptando a lo largo de su evolución, diversas técnicas de aprendizaje de máquina para la detección de tráfico malicioso.

Como se presenta en el capítulo 2, existe una gran cantidad de investigaciones relacionadas al análisis de modelos de Aprendizaje Profundo para la detección de intrusos de red, pero entre estas investigaciones no se encuentran estudios que permitan determinar qué modelo de aprendizaje profundo tiene un mejor desempeño de detección de tráfico de red malicioso, utilizando para este propósito un conjunto de datos de referencia actualizado. Esto, considerando que aproximadamente el 75% de los modelos de aprendizaje de máquina relacionados con la detección de intrusos han utilizado un conjunto de datos de referencia creado en el año 1998 (Hindy et al., 2018).

1.1.2. Formulación del problema

La falta de un estudio comparativo que permita determinar cuál modelo de aprendizaje profundo tiene el mejor desempeño para la detección de tráfico de red malicioso, utilizando para la creación del mencionado modelo, un conjunto de datos de referencia actualizado, implica que los investigadores y desarrolladores de sistemas de detección de intrusos de red no tengan una visión clara sobre el modelo más óptimo que puede ser implementado en un ambiente real para la protección de una red de datos.

1.2. Objetivo general

Realizar un estudio comparativo que determine cuál de los modelos de aprendizaje profundo, desarrollados utilizando Redes Neuronales Recurrentes y Redes Neuronales Convolucionales, a partir del conjunto de datos CICIDS2017, tiene mayor desempeño para la detección de intrusos de red.

1.2.1. Objetivos específicos

- Analizar los modelos de Redes Neuronales Recurrentes y Redes Neuronales Convolucionales, mediante un estudio documental que determine sus principales características.
- Analizar el conjunto de datos CICIDS2017 mediante un estudio descriptivo que determine sus características e información más relevante.
- Realizar el pre procesamiento del conjunto de datos CICIDS2017 mediante el lenguaje de programación python, para la obtención de los datos de entrenamiento y pruebas.
- Codificar mediante el lenguaje de programación python, los modelos de Redes Neuronales Recurrentes y Redes Neuronales Convolucionales antes analizadas, para su posterior entrenamiento en un modelo de aprendizaje profundo.
- Crear los modelos de predicción, mediante el entrenamiento de las redes neuronales antes mencionadas, consiguiendo que ambos modelos converjan y se estabilicen en base a los datos de entrada.

- Validar los modelos de predicción mediante técnicas de validación cruzada con datos reales, para la determinación de la solidez de dichos modelos de predicción.
- Realizar el análisis descriptivo de los resultados obtenidos de la validación de ambos modelos, para la identificación del modelo con mejor desempeño en la detección de intrusos de red.

1.3. Justificación

Con la revisión de la literatura de los trabajos realizados por varios autores relacionados a los objetivos de este tema de investigación, no se ha encontrado un estudio comparativo que permita determinar cuál de los modelos de Redes Neuronales Recurrentes y Redes Neuronales Convolucionales tiene un mayor desempeño en la detección de intrusos de red, mediante la clasificación del tráfico de red en *maligno* y *benigno*.

En los estudios revisados se encontró una gran predominancia en el uso del conjunto de datos KDD-99, para el entrenamiento y pruebas de modelos de aprendizaje de máquina aplicados a los sistemas de detección de intrusos de red. Como se presenta más adelante, el conjunto de datos KDD-99, fue un conjunto de datos preparado en 1998 por Laboratorios Lincoln, el objetivo de este conjunto de datos era simular diversas intrusiones en un entorno de red militar.

Pero, con el exponencial crecimiento del tamaño de las redes de computadoras y el desarrollo acelerado de nuevas aplicaciones, se puede creer que utilizar un conjunto de datos que fue creado hace más de 20 años, no es una opción viable para este tipo de estudios, por lo que para esta investigación se propone utilizar un conjunto de entrenamiento y pruebas actualizado y más apegado a la realidad de las redes de datos en nuestros días, el conjunto de datos CICIDS2017, desarrollado por el Instituto Canadiense para la Ciberseguridad, el cual contiene tráfico de red benigno y con los ataques de red comunes más actualizados capturado en un entorno de red real.

1.4. Marco Teórico

En esta sección se detalla los principales conceptos acerca del desarrollo de la solución de esta investigación, como son: la seguridad informática, las amenazas informáticas, detección de intrusos y las diversas técnicas para su detección. Posteriormente, se analiza las redes neuronales artificiales y los dos tipos de redes neuronales a las que está sujeta esta investigación, y finalmente se aborda conceptos básicos de aprendizaje profundo (*Deep Learning*).

1.4.1. Seguridad Informática

Según Aguilera (2011), la seguridad informática es el área que se encarga de diseñar normas, procesos, métodos y técnicas que permiten obtener un sistema de información seguro, confiable y mantenga una alta disponibilidad.

La seguridad informática busca minimizar los riesgos del manejo de la información en un sistema, contemplando de manera general tres actores (Romero et al., 2018): los usuarios, la información y la infraestructura. Los usuarios son considerados el eslabón más débil de la cadena, ya que son los más propensos a cometer errores. La información como el activo más importante de una organización, es lo que la seguridad informática busca proteger ante una eventual amenaza. Y la infraestructura, un medio más controlado pero no menos riesgoso, es en la que puede encontrar problemas como accesos no permitidos, robos de identidad o daños físicos de la misma.

Dependiendo de la fuente de la amenaza, se distinguen tres tipos de seguridad informática (Díaz & Salcedo, 2014):

- Física, la que busca proteger de un ataque directo, ya sea sobre el usuario, la información o la infraestructura, implementando barreras tangibles y procedimientos de control como medida de prevención y contramedidas.
- Ambiental, la cual busca implementar procedimientos para controlar que los factores ambientales no afecten los equipos de la red o al personal técnico.

- Lógica, en la cual se implementan procedimientos para el control de accesos lógicos a la información, ya sea a la información que se encuentra almacenada o durante su transmisión.

Como se mencionó, la seguridad informática busca minimizar los riesgos, los cuales son considerados como una eventualidad que interrumpe o imposibilita el cumplimiento de un objetivo. Los riegos pueden ser clasificados en cuatro (Díaz & Salcedo, 2014):

- Revelación, que es el acceso no autorizado a información.
- Engaño, que la admisión de datos falsos o irreales.
- Perturbación, que es la interrupción del correcto funcionamiento de un sistema.
- Usurpación, que es el control no autorizado de una parte o de todo un sistema.

1.4.2. Amenazas Informáticas

Se puede definir a una amenaza como la fuente o causa de un evento o incidente no deseado que pueda resultar en el daño o pérdida de un recurso informático de la organización. Entre ellas se identifican las siguientes (Quiroz & Macías, 2017):

- Malware, programa cuyo objetivo es infiltrarse en un sistema, con el objetivo de causar daño en el comportamiento del sistema.
- Pérdida, destrucción, alteración, sustracción, divulgación de información por parte del personal de la organización, debido a mala capacitación, malas prácticas, irresponsabilidad laboral, ignorancia y/o apagado o falta de dispositivos de seguridad.
- Pérdida, destrucción, alteración, sustracción, divulgación de información por parte personas externas malintencionadas.
- Acceso no autorizado a información.
- Ataque de denegación de servicios (*DoS Attack)*, busca dejar fuera de servicio un recurso o sistema específico de una red a través de la sobre carga del mismo con requerimientos.

- Ataque de denegación de servicios distribuido (*DDoS Attack)*, ocurre cuando múltiples sistemas buscan inundar el ancho de banda de un recurso o sistema de una víctima.
- *Phishing*, ataque que busca el robo de identidad y/u otros tipos de ataque de ingeniería social.
- *Spam*, difusión de información de carácter publicitario que saturan o desperdician los canales de comunicación de las organizaciones.
- Pérdida o destrucción de información debido a accidentes o fallas en los equipos.
- Pérdida o destrucción de información debido a catástrofes naturales.
- Ataque de fuerza bruta (*Bruce force attack*), es una popular forma de ataque utilizado para romper contraseñas, y encontrar información oculta en aplicaciones web (Dave, 2013).
- *Heartbleed Attack*, este ataque surge a partir de un *bug* o fallo en la librería de criptografía *OpenSSL*, en la implementación del protocolo TLS (*Transport Layer Security*) (Durumeric et al., 2014).
- *Botnet*, un conjunto de dispositivos conectados a través de Internet, utilizados para diversas actividades como robo de información o envió de correo no deseado (Silva, Silva, Pinto, & Salles, 2013).
- *SQL Injection*, es un tipo de ataque contra un sitio o aplicación web en el cual se busca añadir una porción de código SQL en un campo de entrada de un formulario web con el objetivo de acceder o modificar los datos (Kindy & Pathan, 2011).
- *Cross-Site Scripting (XSS)*, en el cual el atacante busca inyectar código malicioso en aplicaciones web que no fueron correctamente probadas, para comprometer la relación de confianza entre un usuario y la aplicación web (Garcia & Navarro, 2009).
- *Infiltration Attack*, este tipo de ataque se produce cuando un agente externo logra explotar una vulnerabilidad de una aplicación de software, logrando acceso al computador de la víctima. Mediante este acceso es posible realizar otro tipo de ataques sobre la red como escaneo de puertos y enumeración de

servicios (Liu, Shi, Cai, & Li, 2012).

1.4.3. Detección de Ataques Intrusos

Los primeros estudios e investigaciones referentes a la detección de intrusos comenzaron en el año 1980, mediante un trabajo de consultoría realizado por el gobierno de los Estados Unidos por Anderson (1980). El mencionado investigador introduce el término amenaza con relación a la seguridad informática, y lo define como la posibilidad de un intento intencional de acceso a la información, manipulación de la información o inutilizar un sistema. Anderson además presentó la idea de que es posible definir el comportamiento normal del usuario de una red mediante el análisis de su actividad en registros de auditoria, de esta manera podrían descubrirse actividades anómalas en la red (Luna, 2015).

Según Stroud y Powell (2003), un intrusión puede definirse como un fallo operacional mal intencionado, que fue inducido en un sistema por agentes internos o externos. Mientras que la detección de intrusiones, según Mohammed & Pathan (2013), puede definirse como el proceso de monitoreo de eventos en un sistema informático o red, con el objetivo de analizar dichos eventos en busca de signos de intrusiones.

Los sistemas de detección de intrusos (*Intrusion Detection System - IDS*) son una de las herramientas empleadas dentro de la seguridad informática, para evitar que la información sea comprometida. Su funcionamiento básico comprende el análisis pormenorizado del tráfico de la red, comparándolo con firmas de ataques conocidos, o buscando comportamientos anormales. Por ejemplo: escaneos sobre la red, paquetes malformados, entre otros. Los sistemas de detección de intrusos no analizan simplemente el tipo de tráfico, sino también revisan el contenido del mismo y como este se comporta en la red. (González, 2003).

Por lo general, los IDS trabajan en conjunto con otras herramientas de defensa como los cortafuegos (*firewall*). Pero se debe considerar que estos pueden llegar a ser

vulnerados, debido a que el cortafuegos filtra el tráfico de red en base a sus encabezados y protocolos, pero no analiza en detalle el contenido de cada paquete (González, 2003).

Los sistemas de detección de intrusos son una segunda línea de defensa, tras el cortafuegos, el cual recibe los paquetes filtrados de este último, y, realiza un análisis de los mismos, buscando determinar que paquetes pueden llegar a comprometer la seguridad de la red. En la actualidad ciertos cortafuegos, tienen incorporado algunas funcionalidades de un IDS, que les permiten complementar su filtrado, aumentado su eficiencia. Aunque estos por cuestión de retardos que se introducen en la entrega de paquetes, no realizan un análisis a profundidad, sino más bien una revisión rápida, dejando la revisión profunda al filtrado del IDS (Luna, 2015).

1.4.4. Técnicas para la Detección de Intrusos

Como se mencionó los *IDS*, tienen como objetivo el detectar si existe o no una intrusión dentro un sistema, basado en criterios de análisis como los comportamientos anómalos. Esta detección es posible realizarla haciendo uso de distintas técnicas que pueden ser clasificadas en tres categorías principales:

- Métodos estocásticos.- permiten modelar el sistema en base a un perfil estadístico, recurriendo para esto a medidas del tráfico en la red, como por ejemplo número de paquetes de un determinado protocolo, número de conexiones, número de paquetes totales circulantes en la red, entre otros. En función de estas medidas, definir un rango de valores que se consideren normales dentro del sistema. Una vez determinados estos rangos, se puede calcular o definir un grado de desviación para cada rango, y generar las alertas correspondientes cuando se detecta un comportamiento anómalo en el tráfico de la red (Salazar, 2016).
- Métodos basados en especificaciones.- este tipo de IDS se fundamentan en la especificación de un conjunto de reglas por parte de un experto que administre la herramienta. La eficiencia del modelo dependerá completamente de la

experticia del administrador al generar las reglas para especificar si un comportamiento es legítimo o no (García, 2015).

- Métodos basados en aprendizaje.- esta técnica permite establecer un perfil normal de comportamiento del sistema, mediante la utilización de técnicas de aprendizaje de máquina (*machine learning*). Creando un modelo que tendrá la capacidad de determinar cuándo un comportamiento escapa del perfil aprendido (García, 2015).

1.4.5. Aprendizaje de máquina

Según Mitchell (1997), el aprendizaje de máquina (*machine learning*) es un campo que estudia la creación de programas de computadora, que mejoran su desempeño en base a la experiencia, es decir, que son programas capaces de cambiar su comportamiento de manera autónoma, gracias a los resultados que van obteniendo. Los programas de computadora que utilizan aprendizaje de maquina han demostrado gran utilidad en diversos campos de aplicación. Uno de ellos por ejemplo es la analítica de grandes volúmenes de datos, en búsqueda de patrones implícitos en los mismos, difíciles de percibir por el ser humano. Estos patrones pueden ser posteriormente extraídos para su análisis, o en su defecto, pueden ser utilizados para ajustar las acciones de los programas de computadora.

Dentro del aprendizaje de máquina, se pueden distinguir dos tipos de aprendizaje: el supervisado y el no supervisado. En el aprendizaje supervisado el algoritmo es entrenado con un conjunto de datos pre definido, logrando de esta manera alcanzar una conclusión o resultado acertado cuando el algoritmo procese un nuevo conjunto de datos con similares características al utilizado durante el entrenamiento. Y en el aprendizaje no supervisado, se busca que el algoritmo sea capaz de encontrar por sí mismo patrones y relaciones dentro del conjunto de datos (Matich, 2001).

Entre los algoritmos o técnicas de aprendizaje de máquina más importante tenemos:

- Redes Bayesianas.- permiten la creación de modelos bayesianos o probabilísticos para determinar la dependencia entre un conjunto de variables aleatorias, que indiquen la probabilidad de la ocurrencia de un determinado evento (Wagner & Dean, 2001). En el caso de la detección de intrusos las redes bayesianas permiten clasificar de manera probabilísticas los tipos de ataques para los cuales fue entrenada previamente la red (Gordo, Gutiérrez, & Rodríguez, 2013).
- Modelos de Markov.- son modelos que representan la probabilidad de que un estado pueda pasar a otro, en el cual, el nuevo estado depende completamente del anterior. En la detección de intrusos este modelo permite representar la transición entre eventos condicionados y determinar la existencia de una anomalía (Wagner & Soto, 2002).
- Lógica Difusa.- es un sistema lógico con criterios flexibles que representa matemáticamente la incertidumbre y la vaguedad (González, 2011). En la detección de intrusos es útil ya que permite trabajar con características ordinales y lineales, además de trabajar con procesos difusos (Salazar, 2016).
- Algoritmos genéticos.- son modelos adaptativos, usados por lo general en problemas de búsqueda y optimización de parámetros. Se fundamentan en reglas empíricas basadas en los principios de evolución natural y genética. Son empleados para la detección de intrusos como clasificadores de lo bueno o malo de la información, basados en la posible conducta de atributos como tipos de protocolos, entre otros (Gestal, 2013).
- Redes neuronales artificiales.- son redes de neuronas, análogas a las redes de neuronas biológicas, que poseen cierta capacidad de procesamiento. Estas redes para la detección de intrusos pueden ser entrenadas con comportamientos normales o anormales de la red, y a partir de este modelo, detectar ataques o comportamientos inusuales que difieran de los patrones iniciales de entrenamiento (Salas, 2005).

1.4.5.1. Redes Neuronales Artificiales

Según Salas (2005) una red neuronal artificial (*ANN*) se define como un esquema de computación distribuida o computación en malla, que se inspira en la estructura del sistema nervioso de un ser humano. En su forma más básica, una red neuronal contiene múltiples procesadores elementales conectados, conocidos como neuronas, que forman un sistema adaptativo el cual mediante un algoritmo de aprendizaje es capaz de ajustar los pesos sinápticos[1] para alcanzar los requerimientos de desempeño de un problema dado. El proceso mediante el cual la red neuronal ajusta los pesos sinápticos para lograr un determinado objetivo, se denomina entrenamiento o aprendizaje, y el procedimiento para llegar a este cometido se denomina algoritmo de aprendizaje o entrenamiento.

Cada neurona que forma una red, tiene cuatro elementos básicos (Serrano, Soria, & Martín, 2010):

- Un conjunto de conexiones, pesos o sinapsis que determinan el comportamiento de la neurona.
- Un sumador, el cual se encarga de sumar todos los pesos que han sido multiplicados por el valor de cada peso o sinapsis de la conexión.
- Una función de activación no lineal, que permite limitar la amplitud de la señal de salida de cada neurona.
- Y, un umbral que determina cuando se activa la neurona.

[1] Pesos sinápticos, se define como la fuerza de conexión sináptica entre dos neuronas, estos pesos pueden ser valores positivos, negativos o cero. Cuando un peso es positivo la neurona se excita, cuando un peso es negativo la neurona se inhibe y cuando es cero no existe comunicación entre las neuronas. Mediante la ajuste de estos pesos la red es capaz de adaptarse a un entorno y tarea (Toral, 2017).

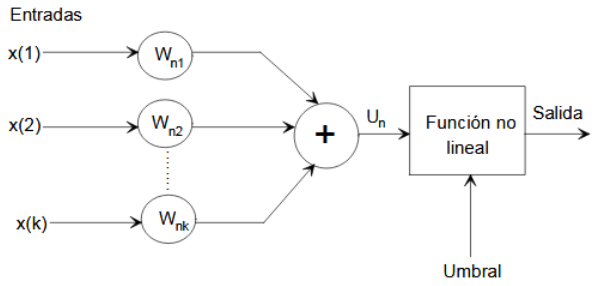

Figura 1. Esquema de modelo Neurona (Serrano et al., 2010)

Las *ANN* tienen la propiedad de aprender a partir de un conjunto de patrones o datos de entrenamiento, es decir, tienen la capacidad de encontrar un modelo que ajuste los datos. En esta etapa de aprendizaje, el modelo busca minimizar el error entre la salida obtenida por la red y la salida deseada por el usuario ante un conjunto de datos de entrenamiento. Este proceso de entrenamiento puede ser supervisado o no supervisado (Salas, 2005):

- El proceso de aprendizaje supervisado consiste en entrenar la red neuronal a partir de un conjunto de datos de entrada y sus respectivas salidas, el algoritmo de aprendizaje ajusta los parámetros de la red de tal manera que la salida generada por la *ANN* se ajuste a los datos de salida dada una cierta entrada. Se dice que es supervisado ya que se conoce desde el inicio cual es el patrón de salida.
- El aprendizaje no supervisado, en cambio, solamente entrega a la *ANN* un conjunto de datos de entrada, y el objetivo del algoritmo de aprendizaje es ajustar los pesos de la red hasta que este encuentre alguna estructura de los datos de entrada.

Las neuronas que conforman una red neuronal, forman distintas arquitecturas de conexión, que pueden clasificarse de la siguiente manera (Serrano et al., 2010):

- Según el número de capas:
    - Monocapa, tiene una capa de neuronas que proyectan las entradas a una capa de neuronas de salida la cual realiza los diferentes cálculos.

    - Multicapa, una generalización de la anterior la cual posee un conjunto de capas intermedias entre la entrada y la salida, llamadas capas ocultas.
- Según el tipo de conexiones:
    - No Recurrentes, en estas redes la propagación de las señales se realiza en un solo sentido, no existe retroalimentación.
    - Recurrentes, estas redes poseen lazos de retroalimentación que pueden ser entre neuronas de diferentes capas, de la misma capa o de la misma neurona.
- Según el grado de conexión:
    - Totalmente conectadas, las neuronas de una capa tienen conexión total con las neuronas de la siguiente capa (no recurrente) o de la capa anterior (recurrente)
    - Parcialmente conectadas, no existe conexión total entre neuronas de las diferentes capas.

1.4.5.2. Redes Neuronales Recurrentes (Recurrent Neural Networks - RNN)

Como se observó anteriormente, las redes recurrentes son un tipo de red neuronal dinámica, ya que el cálculo de una entrada, en un paso determinado depende del paso anterior, y ciertos casos dependen de un paso futuro.

Esencialmente este tipo de redes neuronales son utilizadas para el tratamiento de secuencias, análisis de trayectorias, predicciones no lineales y modelación de sistemas dinámicos (Cruz et al., 2007).

Dependiendo del grado de conexión de las neuronas que conforman las redes recurrentes, estas pueden clasificarse en dos (Cruz et al., 2007):

- Parcialmente recurrentes, donde las conexiones recurrentes son fijas. Conexiones que generalmente son hacia adelante, pero se incluye un conjunto de conexiones de retroalimentación.
- Totalmente recurrentes, donde cada neurona puede estar conectada a cualquier

otra neurona (*full mesh*) y las conexiones recurrentes son variables.

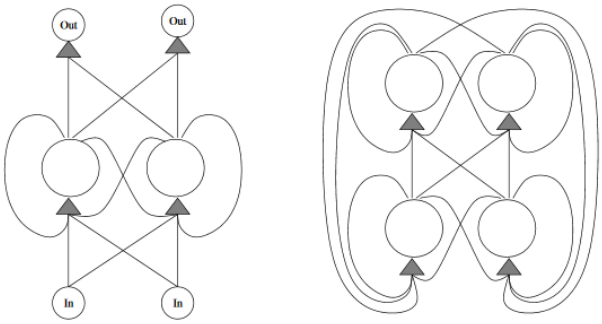

Figura 2. Izquierda: Red Parcialmente Recurrente Derecha: Red Completamente recurrente (Gers, 2001)

Tomando en cuenta solamente las redes parcialmente recurrentes, según Lin & Lee (1996), estas poseen dos tipos de modelos: la red simple recurrente de Elman (1990) y la red secuencial de Jordan (1986). Ambos tipos de son utilizados en la modelización de series temporales.

Mientras que en las redes totalmente recurrentes, se destacan los modelos: RTRL (*Real-Time Recurrent Learning*), TDRB (*Time-Dependent Recurrent Back Propagation*) y LSTM (*Long short-term memory*). Los dos primeros modelos antes mencionados son utilizados generalmente para aplicaciones que requieren aprendizaje en vivo, típicamente utilizados en el ámbito industrial.

El modelo LSTM es una arquitectura particular de una red totalmente recurrente, propuesta por Hochreiter y Schmidhuber (1997), que surge para solventar algunas dificultades presentadas en las redes totalmente recurrentes RTRL y TDRB (Gers, 2001).

Las RNN tienen dos métodos de entrenamiento: la retropropagación y el aprendizaje recurrente a tiempo real. El primer método es un algoritmo de aprendizaje en el cual, un conjunto de datos de entrada se propaga a través de la red neuronal hasta la capa de salida, una vez en este punto se calcula el error producido, y el error es transmitido hacia atrás a las neuronas de las capas ocultas, en base a este error la red neuronal cambia los pesos de las conexiones en cada neurona. Este método utiliza el

gradiente descendiente para definir la función que proporciona el error que comete la red neuronal, ante un determinado conjunto de pesos sinápticos. El objetivo de este entrenamiento es encontrar la configuración de pesos sinápticos que corresponda al mínimo global de la función de error (Bertona, 2005).

El segundo método de entrenamiento tiene una complejidad de cálculo mucho mayor a la que presenta el primer método. Este método tiene la capacidad de ajustar los pesos sinápticos de la red en tiempo real, es decir, mientras la red continua realizando su función de procesamiento. De manera general, este método hace uso del gradiente descendiente para calcular la función de error en cada instante del tiempo.(Cheriyan, Nandakumar, & Sindhu, 2010).

1.4.5.2.1. Long Short-Term Memory (LSTM)

Es un tipo específico de arquitectura de RNN, diseñado para modelar secuencias temporales en dependencias de largo alcance, más precisamente que las RNN convencionales. Las redes LSTM tienen unas unidades especiales llamadas bloques de memoria en la capa recurrente oculta. Cada uno de estos bloques de memoria contiene una celda de memoria con auto-conexiones que almacena el estado temporal de la red, además cuentan con unidades multiplicativas llamadas *gates* que controlan el flujo de información de la red. Cada bloque de memoria tiene un *gate* de entrada y de salida (S. Liu, Hu, & Wang, 2018).

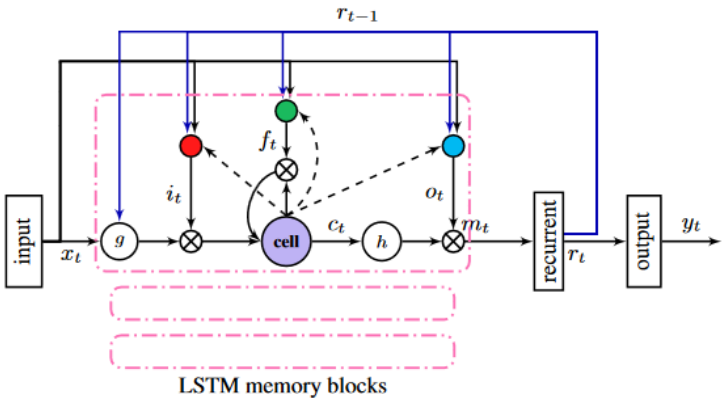

Figura 3. Bloque de memoria LSTM (S. Liu et al., 2018)

La arquitectura LTSM aborda el inconveniente que poseen las RNN convencionales en dependencias a largo plazo, debido a problemas con el gradiente de la red en cálculo del error. Este problema se produce ya que en redes profundas como las RNN, los gradientes de la red son calculados como un producto de diferenciales, entrenar estas redes a lo largo del tiempo, puede llevar al gradiente a desaparecer (*Vanishing*) o acumular grandes gradientes de error lo que dan como resultado actualizaciones de pesos muy grandes (*Exploding*) (Grosse, 2017).

1.4.5.3. Redes Neuronales Convolucionales (Convolutional Neural Networks - CNN)

Las redes neuronales convolucionales, son redes neuronales unidireccionales, donde la información fluye en un solo sentido, desde la capa de entrada hacia la de salida. Son análogas a las ANN tradicionales, en el sentido en que cada neurona de la red se optimiza en base al aprendizaje recibido. Cada neurona obtiene una entrada, y ejecuta una operación.

Una CNN es una red multicapa que consta de: un conjunto de capas convolucionales, un conjunto de capas de pooling o submuestreo y un conjunto de capas totalmente conectadas, similar a una red de perceptrones multicapa (Zaragoza, 2018).

Capa Convolucional

Esta capa lleva a cabo operaciones de producto y sumas entre los datos de entrada y un filtro, generando de esta manera un mapa de características. Cada filtro es capaz de extraer la misma característica en cualquier parte del conjunto de datos de entrada, el filtro se va desplazando a través de la conjunto de datos, hasta obtener la matriz de activación completa que contiene las características que se buscan en el conjunto de datos para cada filtro (Zaragoza, 2018).

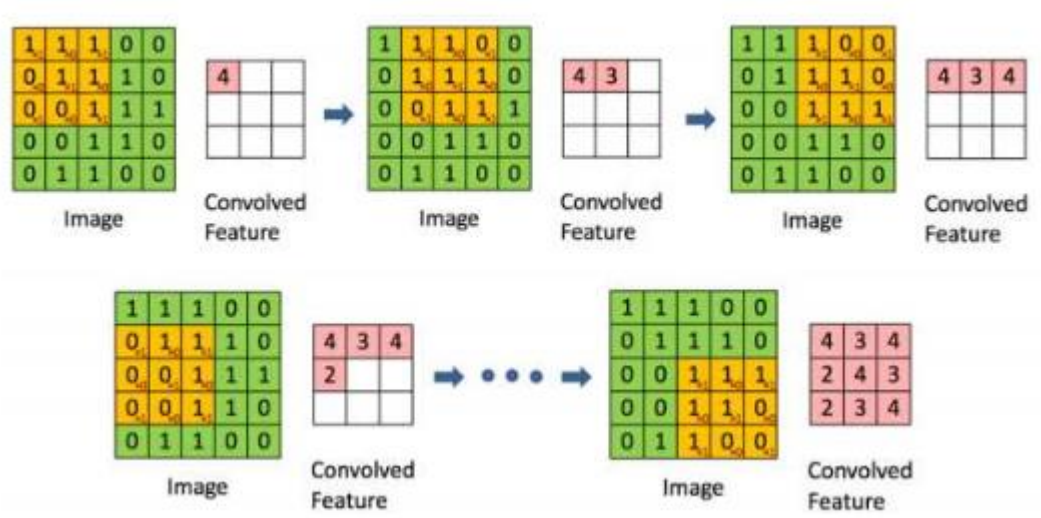

Figura 4. Obtención de Matriz de Activación (Durán, 2017)

Para la presente investigación, se utilizó un tipo de capa convolucional 1D, constituida por una sola dimensión espacial. Este tipo de capa convolucional es muy efectiva para obtener características de segmentos de longitud fija y donde la ubicación de la características en el segmento no sea muy relevante. Las capas convolucionales 1D son muy utilizadas en el análisis de secuencias de tiempos de los datos obtenidos por sensores (Zaragoza, 2018).

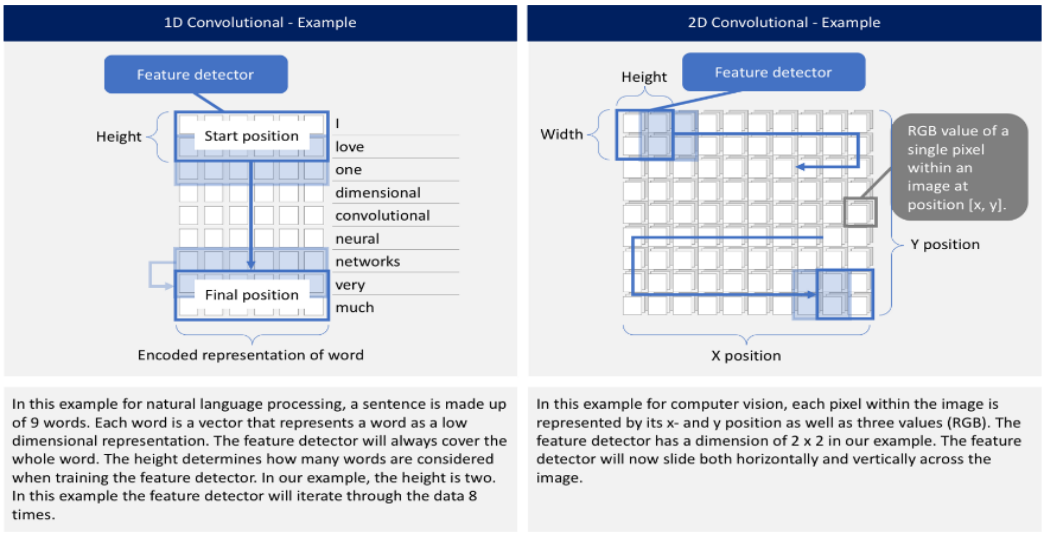

Figura 5. Diferencia entre las capas convolucionales 1D y 2D (Ackermann, 2018)

Capa Pooling

Una vez se ha obtenido las características de los datos de entrada, se procede con la clasificación de los datos en base a estas características, pero debido a la cantidad de características que se pueden obtener, esta clasificación seria computacionalmente muy costosa. La capa de *pooling* ayuda a disminuir la dimensionalidad de las matrices de activación, buscando obtener el máximo valor de una característica a lo largo de una región del conjunto de datos. Con esto disminuimos la carga computacional de todo el sistema y se obtienen rasgos predominantes de los datos de entrada (Durán, 2017).

Esta capa funciona de la siguiente manera, en la esquina superior de la matriz de activación obtenida mediante la capa convolucional, seleccionamos una sub matriz de tamaño $p \ x \ p$. Esta sub matriz, dependiendo del tipo de *pooling*, disminuye la dimensionalidad de la matriz de activación de dos maneras:

- A*verage-pooling*, toma todos los elementos de la sub matriz, calcula su media y se guarda el valor en la matriz de salida.
- M*ax-pooling*, busca el mayor valor que se encuentre en la sub matriz y lo guarda en la matriz de salida.

Posterior a esto se selecciona la siguiente sub matriz que está a $p + 1$ posiciones a la derecha y se realiza la misma operación, hasta recorrer toda la matriz de activación (Durán, 2017).

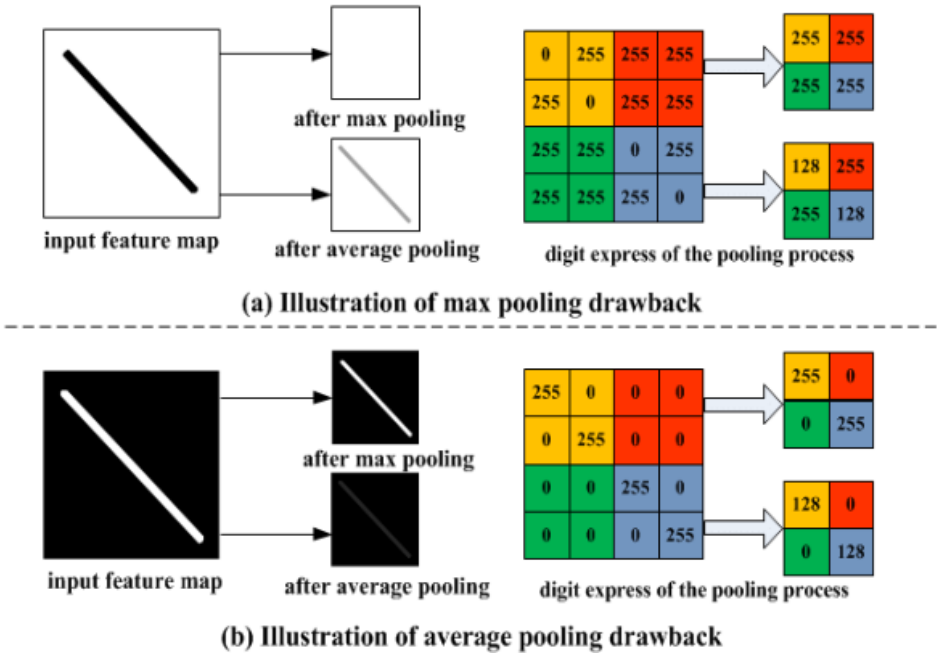

Figura 6. Aplicación de diferentes tipos de Pooling (Durán, 2017)

Capa Totalmente Conectada

Es la última capa de la red convolucional, y se trata básicamente de un clasificador, colocado a continuación de la capa de *pooling*, y que permite en base a las características extraídas anteriormente determinar a qué clase pertenece la entrada (Zaragoza, 2018).

1.4.6. DEEP LEARNING

Son un conjunto de técnicas y procedimientos basados en *Machine Learning*, que permiten el entrenamiento o aprendizaje por niveles. Tiene la capacidad de entrenar cada capa de la arquitectura del modelo, tomando las características precedentes de la capa anterior como entrada para el entrenamiento de la siguiente capa (Du & Shanker, 2013).

Se emplea habitualmente para el entrenamiento no supervisado de redes neuronales multicapa. Estos algoritmos de aprendizaje de máquina utilizan métodos computacionales para lograr aprender información (características abstractas y de alto

nivel) directamente del conjunto de datos, sin depender de una ecuación predeterminada como modelo (González, 2018).

De lo mencionado anteriormente se puede indicar que, la diferencia principal entre *Machine Learning* y *Deep Learning*, se encuentra en la fase conocida como *feature engineering.* Esta fase mapea los datos de entrenamiento a una salida determinada acorde la función del modelo, para posteriormente replicarse sobre nuevos datos de entrada, para obtener otra salida acorde a la función del modelo (Carreras, 2018).

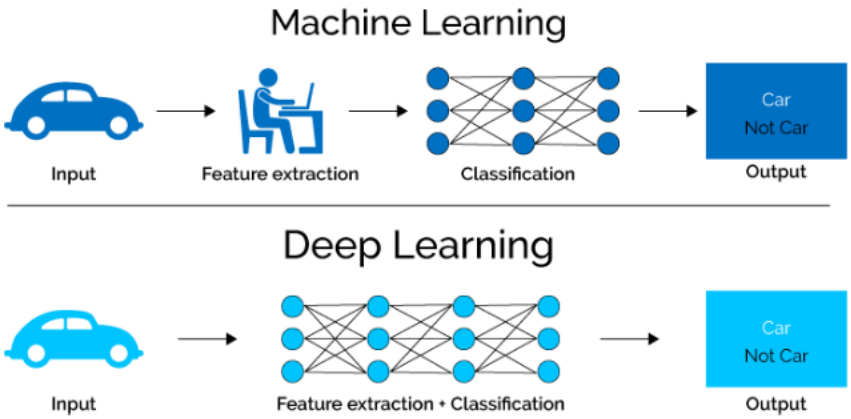

Figura 7. Comparación entre machine learning y deep learning (Carreras, 2018)

Los modelos de *Deep Learning* están construidos mediante redes neuronales simples que combinadas forman redes neuronales profundas. Este tipo de técnicas han permitido un gran avance en el campo del procesamiento de sonidos e imágenes, visión artificial, procesamiento automático de lenguaje, clasificación de textos, tecnologías avanzadas de asistencia al conductor, entre muchas otras potenciales aplicaciones (Du & Shanker, 2013).

CAPÍTULO II

ESTADO DEL ARTE

La detección de intrusos juega un papel muy importante en el proceso de defensa en una red de datos, ya que ayuda a los administradores de seguridad a detectar comportamiento anómalo o malicioso como intrusiones, ataques y malware. Los sistemas de detección de intrusos se han convertido en un requisito casi indispensable en redes críticas que permitan hacer frente a las actividades maliciosas de los intrusos. Es así que la investigación sobre estos sistemas ha progresado a lo largo de los años para proponer mejores IDS.

2.1. Aprendizaje de máquina para detección de intrusiones

2.1.1. Aprendizaje de máquina para detección de malware

El *malware* según Bayer, Moser, Kruegel, & Kirda (2006), es un programa o *software*, que deliberadamente cumple con la intención dañina de un atacante. Este software está diseñado para obtener acceso a sistemas informáticos o a recursos de red, interrumpir o perturbar el normal funcionamiento de un sistema o recopilar información sin el consentimiento del propietario de un sistema. Tiene un amplio rango de variaciones como virus, gusanos, troyanos, *Rootkit*, puertas traseras, *botnets*, *spywares*, *adwares*, etc. El *malware* se ha convertido en una de las principales amenazas de seguridad que enfrenta Internet estos días (Idika & Mathur, 2007).

La detección de *malware* mediante el uso aprendizaje de máquina fue introducido por primera vez por Schultz, Eskin, Zadok, & Stolfo (2002), ellos usaron tres diferentes de características para la clasificación de malware: *Portable Executable* (PE), cadenas de caracteres y secuencia de bytes. Para su investigación utilizaron un conjunto de datos de 4266 archivos (3265 maliciosos y 1001 benignos).

Kolter & Maloof (2004) utilizaron tres clasificadores como las redes bayesianas, *support vector machines* (SVM) y árboles de decisión para la detección de ejecutables maliciosos, estos algoritmos buscaban encontrar patrones en las secuencias de bytes de los ejecutables. Utilizaron 1971 programas benignos y 1651 malignos. Llegaron a la conclusión que el algoritmo de árboles de decisión obtuvo el mejor resultado de clasificación con un 98% de acierto.

Lee & Mody (2006), propusieron un método en el cual a través de algoritmos de aprendizaje de máquina, agruparon muestras de *malware* que fueron ejecutados en un ambiente virtual y fueron monitoreadas las llamadas al sistema que realizaba cada muestra. Crearon perfiles de comportamiento de cada muestra en base a la información recogida de la interacción de cada muestra. De esta manera se agruparon los distintos perfiles y sus respectivas muestras.

Tian, Batten, & Versteeg (2008)**,** utilizaron una función de frecuencia de longitud para clasificación de troyanos. La función de longitud mide el número de bytes totales del código del *malware*. Los resultados de la investigación mostraron que la función de longitud junto con la frecuencia de aparición es significativa para la identificación de *malware,* y que junto con otras características se puede lograr una identificación más rápida y escalable.

Siddiqui, Wang, & Lee (2008), utilizaron la variable de la longitud de la secuencia de instrucción para la clasificación de gusanos mediante tres tipos de clasificadores (árboles de decisión, *bagging* y *random forest*), utilizaron 1444 gusanos y 1330 archivos benignos. Obtuvieron un 95,6% de detección a través del algoritmo *random forest*.

Tian, Batten, Islam, & Versteeg (2009), utilizaron cinco algoritmos de aprendizaje de maquina (redes bayesianas, *support vector machines*, IB1, *random forest* y árboles de decisión) disponibles en el software WEKA, para realizar clasificación de *malware*, sobre 1367 muestras. Obtuvieron un 97% de precisión.

Rieck, Trinius, Willems, & Holz (2011), propusieron un método para el análisis automático del comportamiento del *malware* utilizando aprendizaje automático. Para esto se recopiló 3133 muestras de malware y supervisó su comportamiento en un entorno aislado. Luego se trasladó el comportamiento observado a un espacio vectorial, y a partir de este aplicaron algoritmos de clusterización y clasificación, se identificaron varias clases de *malware.*

Nataraj, Karthikeyan, Jacob, & Manjunath (2011), propusieron un método para clasificar *malware* mediante el uso de técnicas de procesamiento de imágenes, las cuales visualizaban al binario del *malware* como una imagen en escala de grises. Se utilizó el algoritmo k-NN (*k-nearest neighbors*) para la clasificación.

Kong & Yan (2013), propusieron un método de clasificación basado en grafos. Extraían todas las características de cada muestra de *malware,* en función del grafo, para posteriormente evaluar la similitud de dos muestras, mediante una técnica de medición de distancia discriminatoria que agrupaba las muestras similares en una misma familia, y manteniendo las muestras distintas separadas a una distancia marginal. Posteriormente técnicas de k-NN y *support vector machines* para clasificar las familias de *malware* en base a las distancias de cada familia.

Gibert & Bejar (2016) propusieron transformar los archivos maliciosos binarios en imágenes en escala de grises 2D y clasificarlos mediante redes neuronales convolucionales 2D. También sugerían tratar el código de operación del *malware* como palabras y clasificarlo mediante una red convolucional 1D. Los autores obtuvieron un 99,52% de precisión en la clasificación.

Wang & Yiu (2016), utilizaron redes neuronales recurrentes, para transformar la secuencia de llamadas API del *malware* en un vector, posteriormente utilizaban un clasificador para determinar la familia del *malware* en base al vector mencionado. Obtuvieron un 99,1% de precisión en la clasificación. Estos autores también propusieron entrenar un segundo clasificado para interpretar patrones de acceso a archivos, para

clasificar ataques de *malware* de día cero, obtuvieron un 99,2% de precisión utilizando una red neuronal recurrente.

2.1.2. Aprendizaje de máquina para detección de red

Inicialmente Anderson (1980) describió por primera vez el concepto de detección de intrusiones, a través de su modelo de clasificación de amenazas basado en la detección de anomalías de comportamiento del usuario.

Denning (1987) propuso un modelo para la detección de intrusos, basado en la estadística, en la cadena de Markov y series de tiempo. Este modelo era independiente del sistema, del ambiente de aplicación, de las vulnerabilidades del sistema o del tipo de ataque, y proveía las bases para los IDS.

Smaha (1988) implementó *Haystack*, un sistema de detección de intrusos desarrollado a partir de restricciones de comportamiento impuestas a través de políticas de seguridad y en modelos de comportamiento típicos de un usuario o un grupo de usuarios. Se estableció un umbral considerado como normal para cada atributo, y una alarma era activada cada vez que algún atributo superaba el umbral. Tenía la capacidad de detectar seis tipos de intrusiones: ingreso de usuarios no autorizados, denegación de servicio, intrusión en el sistema de administración de seguridad, ataques de usuarios enmascarados.

Forrest, Hofmeyr, Somayaji, & Longstaff (1996), propusieron una analogía entre el sistema inmune humano y un sistema de detección de intrusos, en el cual se analizaban las secuencias de las llamadas a un sistema de un programa para crear un perfil normal de comportamiento.

Lee, Stolfo, & Mok (1999) propusieron un modelo adaptativo para IDS, buscaban mediante programas de auditoría, extraer características que describan conexiones de red, sesiones en cada equipo, y utilizando programas de minería de datos aprender reglas que describan el comportamiento de intrusos o comportamiento normal.

En la actualidad los sistemas de detección de intrusos centran su método de detección en un único aspecto, como los algoritmos de aprendizaje de máquina, lo cual ha sido ampliamente desarrollado por investigadores para el diseño de estos sistemas (Hindy et al., 2018). A continuación se presentan las investigaciones más destacadas referentes al uso de algoritmos de aprendizaje de máquina en sistemas de detección de intrusos. Las mismas estarán divididas en tres categorías: *Single Classifiers*, *Hybrid Classifiers* y *Ensemble Classifiers.*

A continuación se presenta una lista de acrónimos que serán utilizados a lo largo del presente capítulo:

Término	Acrónimo
Redes Neuronales Artificiales	ANN
Árboles de Decisión	DT
Redes Bayesianas	NB
K-nearest neighbors	KNN
Algoritmos Genéticos	GA
Random Forest	RF
Red Neuronal Recurrente	RNN
Red Neuronal Convolucional	CNN
Long short-term memory	LSTM
Principal Component Analysis	PCA
Support Vector Machine	SVM

Tabla 1. Lista de acrónimos

Single Classifiers

Esta categoría hace referencia a los estudios en los cuales se utilizó un solo método o algoritmo de aprendizaje de máquina para la detección de intrusos (Akhi, Kanon, Kabir, & Banu, 2019).

Año	Autor(es)	Algoritmos utilizados
2002	Liao & Vemuri	KNN
2003	Giacinto & Roli	-Multiple Classifier System -K classifiers
2003	Heller, Svore, Keromytis, & Stolfo	SVM
2003	Joo, Hong, & Han	ANN para considerar la relación de costos de errores falsos negativos y errores falsos positivos.
2004	Wang, Guan, & Zhang	PCA
2004	Scott	NB
2004	Wang & Stolfo	KNN
2005	W.-H. Chen, Hsu, & Shen	SVM ANN
2005	Zhang & Shen	SVM
2005	Kang, Fuller, & Honavar	-NB -DT -SVM -Regresión logística
2006	Wang & Battiti	PCA
2010	Hou, Chang, Chen, Laih, & Chen	-NB -DT -SVM
2010	Wu & Huang	ANN
2011	Davanzo, Medvet, & Bartoli	-KNN -Local Outlier Factor -Local Hotelling's T-Square

Año	Autor(es)	Algoritmos utilizados
		Parzen windows -SVM -Domain Knowledge Aggregator
2011	Wagner, State, Engel, & Learning	OCSVM (One- class SVM)
2012	Catania, Bromberg, & García	SVM
2012	Kang, Jeong, & Kong	Support Vector Data Description (SVDD)
2012	Kumar, Kaur, & Kumar	C4.5
2013	Sahin, Bulkan, & Duman	DT
2013	Grinblat, Uzal, & Granitto	SVM
2013	Yassin, Udzir, Muda, & Sulaiman	KMC (K-Means Clustering) y NBC (Clasificador basado en redes bayesianas)
2016	Moustafa & Slay	-DT -ANN -Regresión Logística -NB -Clusterización EM (Expectation- Maximization)
2016	Hodo et al.	ANN
2017	Li, Meng, Kwok, & IP	-KNN -Back-propagation neural networks (BPNN) -DT
2017	Hodo et al.	-SVM -ANN

Año	Autor(es)	Algoritmos utilizados
2018	AL-Hawawreh, Moustafa, & Sitnikova	Deep Auto-Encoder

Tabla 2. Investigaciones aprendizaje de máquina (Single classifiers) en sistemas de detección de intrusos.

Hybrid Classifiers

Esta categoría hace referencia a los estudios en los cuales se combinan varios algoritmos de aprendizaje de máquina para mejorar el rendimiento en los sistemas de detección (Akhi et al., 2019).

Año	Autor(es)	Algoritmos utilizados
2000	Luo & Bridges	-Lógica difusa -Minería de datos (reglas de asociación)
2004	Peddabachigari, Abraham, & Thomas	-DT -SVM
2004	Chavan et al.	Neuro-Fuzzy (ANN y Lógica Difusa)
2006	Liu & Yi	PCASOM (ANN)
2007	Liu, Yi, & Yang	-PCA -ANN
2007	Chen, Abraham, & Yang	-Flexible neural tree (FNT)
2007	Saniee, Habibi, Barzegar, & Sergi	Parallel genetic local search algorithm (PAGELS) basado en lógica difusa y GA
2009	Tong, Wang, & Yu	-RBF -Elman ANN
2010	Shim, Kim, & Kim	Clusterización

Año	Autor(es)	Algoritmos utilizados
2012	Gaikwad, Jagtap, Thakare, & Budhawant	FC-ANN basado en Redes Neuronales, logica difusa y clusterización
2013	Shin, Lee, Kim, & Kim	-K-means clustering -Cadenas de Markov
2015	Alipour, Al-Nashif, Satam, & Hariri	Anomaly Behavior Analysis (ABA)
2015	Pan, Morris, & Adhikari	Path Mining algorithm
2017	Balamurugan & Saravanan	-Packet scrutinization algorithm -K-means clustering algorithm -RNN
2017	Tran, Vo, & Thinh	-GPU-based ANN -Back-propagation ANN
2018	Kumar & Sharma	Signature-Based Anomaly Detection Scheme (SADS)
2019	Chawla, Lee, Fallon, & Jacob	CNN RNN

Tabla 3. Investigaciones aprendizaje de máquina (Hybrid classifiers) en sistemas de detección de intrusos.

Ensemble Classifiers

En esta categoría se utilizaron combinaciones de varias técnicas de aprendizaje de máquina en un modelo predictivo para mejorar los resultados de los sistemas de detección (Akhi et al., 2019).

Año	Autor(es)	Algoritmos utilizados
2004	Fan, Miller, Stolfo, Lee, & Chan	Distribution-based artificial anomaly algorithm

Año	Autor(es)	Algoritmos utilizados
2005	Mukkamala, Sung, & Abraham	-Multivariate Adaptive Regression Splines (MARS) -ANN -SVM
2007	Khan, Awad, & Thuraisingham	-SVM -Dynamically Growing Self-Organizing Tree (DGSOT)
2008	Das, Schneider, & Neill	-Likehood NB -Conditional Anomaly Detection -WSARE
2009	Lu & Tong	-Non-Parametric CUSUM -EM based Clustering
2012	Chi Cheng, Wee Peng Tay, & Huang	Método ELM
2013	Fadlullah, Nishiyama, Kato, & Fouda	Non-parametric cumulative sum (CUSUM)
2015	Ng, Joshi, & Banik	Clusterización
2016	Ambusaidi, He, Nanda, & Tan	-SVM -Mutual Information algorithm
2018	Nagamani & Chittineni	RF

Tabla 4. Investigaciones aprendizaje de máquina (Ensemble classifiers) en sistemas de detección de intrusos.

2.1.3. Aprendizaje de máquina para detección de intrusos red utilizando el conjunto de datos de referencia KDD99 y NSL-KDD

El conjunto de datos de referencia KDD99 fue preparado por Stolfo, Fan, Lee, Prodromidis, & Chan (2000), a partir de los datos capturados en el programa de evaluación para IDS desarrollado por la DARPA en el año 1998. Los datos capturados en este programa son 4 gigabytes de tráfico de red recuperado en un periodo siete semanas

(Lippmann et al., 2000). El conjunto de datos KDD99 consta de 4´900.00 vectores de conexión, donde cada vector se encuentra definido por 41 características y una etiqueta que clasifica el tráfico como normal o como ataque, definiendo la clase de ataque especifico, los cuales son: *Denial of Service Attack* (DoS), *User to Root Attack* (U2R), *Remote to Local Attack* (R2L) y *Probing Attack* (Lippmann et al., 2000).

A continuación se presentan las investigaciones más destacadas referentes al uso de algoritmos de aprendizaje de máquina en sistemas de detección de intrusos en los cuales se utilizó como conjunto de referencia el KDD99 para el entrenamiento y pruebas de los algoritmos mencionados, además se indica los resultados obtenidos en cada investigación. Al igual que en el punto anterior las investigaciones están divididas en tres categorías (*Single classifiers, Hybrid classifiers y Ensemble classifiers*).

Single Classifiers

Año	Autor(es)	Algoritmos utilizados	Resultados
2002	Eskin, Arnold, Prerau, & Portnoy	-KNN -SVM Clusterización	Precision KNN 91% SVM 98% Cluster 93%
2004	Moradi & Zulkernine	ANN	Precisión 91% con 2 capas ocultas y 87% con 1 capa oculta
2006	Chimphlee, Abdullah, Sap, Srinoy, & Chimphlee	Lógica difusa	Precisión 82.46% Tasa de detección 91.45% Tasa de Falsas alarmas 24.8%
2007	Yang Li & Guo	KNN	Precisión 99.7%
2007	Gunes, Nur, & Heywood	SOM con aprendizaje no supervisado.	Precisión 9.,4% Tasa de Falsos positivos 1.38%
2009	S.-Y. Wu &	-SVM	Precisión

Año	Autor(es)	Algoritmos utilizados	Resultados
	Yen	-C4.5	SVM Probe 82.7% DOS 60.04% U2R 41.05% R2L 17.46% C4.5 Probe 86.30% DOS 62.96% U2R 50.06% R2L 17.43%
2010	Mok, Sohn, & Ju	Logistic Regression	Precisión 98.68%
2012	Koc, Mazzuchi, & Sarkani	NB	Precisión 93.72% Tasa de error 0.628%
2013	Devaraju & Ramakrishnan	-Feed Forward Neural Network (FFNN) -Elman Neural Network (ENN) -Generalized Regression Neural -Network (GRNN) -Probabilistic Neural Network (PNN) -Radial Basis Neural Network (RBNN)	Eficiencia FFNN 79.49% ENN 78.1% GRNN 58.74% PNN 85.56% RBNN 83.51%
2015	Eesa, Orman, & Brifcani	Cuttlefish Optimization Algorithm (Feature Selection)	Tasa de detección 71.087% Tasa de falsos positivos 17.685% Precisión 73.267%

Año	Autor(es)	Algoritmos utilizados	Resultados
2015	Masduki, Ramli, Saputra, & Sugiarto	SVM	Precisión de detección ataque R2L SVM 8 características sin payload 95.87% SVM 8 características con payload 95.99% SVM 24 características sin payload 95.73% SVM 24 características con payload 95.91% SVM 28 características sin payload 95.91% SVM 28 características con payload 96.08%
2016	Ji, Jeong, Choi, & Jeong	-SVM -ANN -NB	Precisión SVM 98.22% ANN 98.59% RB 94.7%
2016	Moustafa & Slay	-DT -ANN -Regresión Logística -NB -Clusterización EM (Expectation- Maximization)	Precisión DT 92.30% ANN 97.04% RL 92.75% NB 95% EM Cluster 78.06%
2016	Subba, Biswas, & Karmakar	-PCA -SVM -ANN -C4.5 -NB	Precisión SVM 99.13% ANN 96.76% C4.5 96.85% NB 94.56%

Año	Autor(es)	Algoritmos utilizados	Resultados
			Tasa de detección SVM 98.68% ANN 96.13% C4.5 97.23% NB 90.75%
2017	Sumaiya & Aswani	SVM	Precisión 98%
2017	Yin, Zhu, Fei, & He	RNN	Tasa de detección Probe 83.4% DOS 83.49% U2R 11.5% R2L 24.69% Tasa de falsos positivos Probe 2.16% DOS 2.06% U2R 0.07% R2L 0.8%
2018	Shone, Ngoc, Phai, & Shi	-Deep Learning -Non-Symmetric Deep Auto-Encoder Stacked NDAEs	Precisión 89.22% Precision 92.97% Recall 89.22% F-score 90.76% Tasa de falsa alarma 10.78%
2018	AL-Hawawreh, Moustafa, & Sitnikova	Deep Auto-Encoder	NSL KDD Tasa de detección 99% Tasa de falsos positivos 1.8%

Tabla 5. Investigaciones aprendizaje de máquina (Single classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos KDD-99.

Hybrid Classifiers

Año	Autor(es)	Algoritmos utilizados	Resultados
2004	Bouzida & Cuppens	-KNN -DT -PCA	Precisión KNN + PCA 92.22% AD + PCA 92.16%
2005	Depren, Topallar, Anarim, & Ciliz	-DT -SOM	Tasa de detección del 99.90% Tasa de clasificación del 99.84% Tasa de Falsos positivos 1.25%
2005	Stein, Chen, Wu, & Hua	-DT -GA	Tasa de error DT Probe 2.166494% DOS 2.321258% U2R 0.095610% R2L 19.979205% DT+GA Probe 1.670193% DOS 2.222582% U2R 0.100885% R2L 19.945019%
2007	Peddabachigari, Abraham, Grosan, & Thomas	-DT -SVM	Precisión SVM Normal 99.64% Probe 98.57% DOS 99.92% U2R 40% R2L 33.92%

Año	Autor(es)	Algoritmos utilizados	Resultados
			AD Normal 99.64% Probe 99.86% DOS 96.83% U2R 68% R2L 84.19% Hibrido SVM + AD Normal 99.7% Probe 98.57% DOS 99.2% U2R 48% R2L 37.8%
2007	Toosi & Kahani	Neuro-fuzzy (ANN y Lógica Difusa)	Tasa de detección 95.3% Tasa de Falsa alarma 1.9%
2009	Bahrololum, Salahi, & Khaleghi	-ANN -SOM	Precisión Normal 97.87% Probe 71.65% DOS 97% U2R 0% R2L 26.69%
2009	Shafi & Abbass	-GA	Precisión Normal 99.38% Probe 75.36% DOS 96.77% U2R 21.52% R2L 2.80%

Año	Autor(es)	Algoritmos utilizados	Resultados
2009	Tajbakhsh, Rahmati, & Mirzaei	-Association Based Classification (ABC) -Fuzzy Association Rules	Tasa de detección 99.6%
2010	G. Wang, Hao, Mab, & Huang	-FC-ANN (Fuzzy clustering, ANN)	Precisión AD 96.75% RB 96.11% BPNN 96.65% FC-ANN 96.71%
2010	Jawhar & Mehrotra	FCM Clustering	Precisión 99.99% Tasa de falsos positivos 0.01%
2011	Abadeh, Mohamadi, & Habibi	Genetic fuzzy systems (GFSs): -Michigan -Pittsburgh -Iterative rule learning (IRL)	Precision Michigan Normal 49.6% Probe 100% DOS 100% U2R 72.2% R2L 99% Pittsburgh Normal 93.7% Probe 98% DOS 99.7% U2R 17.4% R2L 97.6% IRL Normal 72% Probe 63.7% DOS 98.4%

Año	Autor(es)	Algoritmos utilizados	Resultados
			U2R 100% R2L 88.9%
2011	S.-S. Wang, Yan, Wang, & Liu	-Rule-Based -BON -ART Network	Tasa de detección 90.96% Precisión 91.26% Tasa de Falsos positivos 2.06%
2011	Su	-KNN -GA	Precisión 97.42%
2011	Lee, Kim, & Kim	-SOM -K-means clustering	Tasa de detección promedio 86,4% Tasa de falsos positivos 0.91%
2011	Muda, Yassin, Sulaiman, & Udzir	-K means -NB	Precisión NB 83.19% KM+NB 99.6% Tasa de detección NB 94.7% KM+NB 99.8% Falsa alarma NB 19% KM+NB 0.5%
2012	Li et al.	-K means -SVM -Algoritmo de colonia de hormigas	Precisión 98.6249%
2012	Aneetha & Bose	-ANN (SOM) -K means	Precisión 98.5%
2013	Chandrashekhar & Raghuveer	-Fuzzy C means -Fuzzy ANN (Neuro-Fuzzy y función Radial Basis)	Precisión Probe 97.11% DOS 98.94% U2R 97.80%

Año	Autor(es)	Algoritmos utilizados	Resultados
			R2L 97.78%
2013	Ahmad & Hanapi	-Fuzzy Clustering -ANN	Precisión mayor al 90% Recall mayor al 80%
2013	Lisehroodi, Muda, & Yassin	-ANN -K-Means Clustering	Precisión Normal 99.99% Probe 99.97% DOS 99.99% U2R 99.99% R2L 99.98%
2014	Kim, Lee, & Kim	-C4.5 -SVM	El estudio muestra que el método propuesto posee un mayor desempeño (tiempo de ejecución y utilización de CPU) es superior comparado con Conventional hybrid method, Misuse detection method (DT) y Anomaly detection method (SVM)
2014	Feng, Zhang, Hu, & Huang	-SVM -Algoritmo de colonia de hormigas	Tasa de detección 94.86% Tasa de Falsos positivos 6.01% Tasa de Falsos negativos 1%
2014	Ranjan & Sahoo	Método clusterización K-medoids	Tasa de detección 91.2% Precisión 96.38% Tasa de falsas alarmas 3.2%
2015	Jabez & Muthukumar	Neighborhood Outlier Factor (NOF)	El estudio muestra que el método NOF posee un mayor desempeño (tiempo de ejecución y utilización de CPU) es superior comparado con Back-

Año	Autor(es)	Algoritmos utilizados	Resultados
			Propagation Neural Network, ANN y fuzzy clustering e Hyperbolic Hopfield Neuronal Network
2016	Reza, Miri, & Javidan	-Synthetic minority oversampling technique (SMOTE) -Cluster center and nearest neighbor (CANN)	Precisión 98.99% Tasa de detección 99.56% Tasa de Falsas alarmas 0.557%
2016	Aslahi-Shahri et al.	-SMV -GA	Tasa de falsos positivos 0.973 Tasa de falsos negativos 0.017
2016	Ambusaidi, He, Nanda, & Tan	-SVM -Mutual Information algorithm	KDD99 Precisión Normal 99.79% Probe 99.91% DOS 99.86% U2R 99.97% R2L 99.92% NSL KDD Tasa de detección 98.93% Tasa de falsos positivos 0.28% Exactitud 99.94
2016	Hosseini Bamakan, Wang, Yingjie, & Shi	-SVM -Multiple criteria linear programming (MCLP)	Tasa de detección TVCPSO–MCLP 97.23% TVCPSO–SVM 97.03%

Año	Autor(es)	Algoritmos utilizados	Resultados
2016	Hadri, Chougdali, & Touahni	-PCA -Fuzzy PCA	Tasa de detección PCA Probe 93,1111% DOS 69,5556% U2R 7,6923% R2L 4,5556% FPCA Probe 91,3333% DOS 73,1111% U2R 13,4615% R2L 4,1111%
2016	Nskh, Varma, & Naik	-SVM -PCA	El estudio demuestra que el uso de SVM + PCA aumento la tasa de detección y disminuye el tiempo de detección en comparación con SVM sin PCA
2017	Raman, Somu, Kirthivasan, & Sriram	Hypergraph and Arithmetic Residue-based Probabilistic Neural Network (HG AR-PNN)	Precisión Normal 94.32% Probe 84.32% DOS 95.75% U2R 80.11% R2L 80.62% Recall Normal 96.27% Probe 86.74 % DOS 98.12% U2R 81.24 %

Año	Autor(es)	Algoritmos utilizados	Resultados
			R2L 81.78 %
2017	Puri & Sharma	-SVM -Classification and Regression tree algorithm	Valores Precisión Normal 1 Probe 0.91 DOS 1 U2R 0.43 R2L 0.92 Tasa de detección Normal 0.998 Probe 0.97 DOS 1 U2R 0.922 R2L 0.91 Tasa de Falsos positivos Normal 0.001 Probe 0.0002 DOS 0.0036 U2R 0.00005 R2L 0.0003
2017	Salunkhe & Mali	Classifier Ensemble basado en Logistic Regression, J48, NB	Tasa de detección LR Phf 100% Teardrop 100% Normal 99.9% Smurf 97.7% Loadmodule 71.4%

Año	Autor(es)	Algoritmos utilizados	Resultados
			J48 Phf 81.3% Teardrop 100% Normal 99.9% Smurf 100% Loadmodule 74.2% NB Phf 71% Teardrop 100% Normal 99% Smurf 100% Loadmodule 85.7%
2017	Syarif & Gata	-PSO Binario -KNN	Precisión K=5 Normal 99.57% Probe 94.41% DOS 99.91% U2R 99.77% R2L 99.73% K=10 Normal 99.62% Probe 94.31% DOS 99.88% U2R 99.74% R2L 99.66%

Año	Autor(es)	Algoritmos utilizados	Resultados
			K=15 Normal 99.31% Probe 94.13% DOS 99.89% U2R 99.58% R2L 99.39%
2017	Xu, Chen, Zhang, & Wu	-R-tree -KNN -K-means -SVM	Tamaño datos de entrenamiento 100000 99.53% (Sensibilidad) 120000 99.68% (Sensibilidad) 140000 99.62% (Sensibilidad) 160000 99.70 (Sensibilidad) 180000 99.71% (Sensibilidad) 200000 99.68% (Sensibilidad)
2017	Zhao, Li, Zia, & Zomaya	-PCA -Softmax Regression -KNN	PCA + Softmax 3 features Tasa de detección 84.347% Tasa falsa alarma 5.128% Exactitud 84.999% 6 features Tasa de detección 91.520% Tasa falsa alarma 4.944% Exactitud 84.436% 10 features Tasa de detección 99.312 % Tasa falsa alarma 1.116% Exactitud 84.406%

Año	Autor(es)	Algoritmos utilizados	Resultados
			PCA+KNN 3 dimensiones Exactitud 85.24% 6 dimensiones Exactitud 85.19%
2017	Effendy, Kusrini, & Sudarmawan	-K-means -NB -Information Gain	Tasa de detección Normal 0.953 Probe 0.975 DOS 0.96 U2R 0.998 R2L 0.964
2018	Duan & Xiao	-Lógica Difusa -KNN	Precisión 98.73% Tasa de detección 96.23% Tasa falsa alarma 0.28%
2018	He, Chen, Zou, Pei, & Jiang	Kernel Clustering	El estudio demuestra que el modelo propuesto posee una alta tasa de detección y una baja tasa de falsas alarmas comparadas con SVM y con el algoritmo de kernel clustering
2019	Hsu, Hsieh, Prakosa, Azhari, & Leu	RNN (LSTM) CNN	Precisión RNN 81.29% LSTM 87.53% CNN+LSTM 88.95%

Tabla 6. Investigaciones aprendizaje de máquina (Hybrid classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos KDD-99.

Ensemble Classifiers

Año	Autor(es)	Algoritmos utilizados	Resultados
2003	Shyu, Chen, Sarinnapakorn, & Chang	PCC (Principal Component Classifier)	Precisión 97.87% Recall 98.94% Tasa de falsa alarma 0.92%
2004	Liu, Chen, Liao, & Zhang	Intrusion Detection Based on Genetic Clustering (IDBGC) -GA -Clusterización	El estudio subdividió todo los datos de KDD99 en 5 conjunto de datos, y aplicaron su método de detección a cada conjunto de datos obteniendo la siguiente exactitud D1 68% D2 33% D3 74% D4 44% D5 79% Promedio 59.6%
2007	Özyer, Alhajj, & Barker	-GA -Lógica difusa -Minería de datos (reglas de asociación y clasificación)	Tasa de detección (por tipo de tráfico) Normal 95.8% Probe 54.10% DOS 97.4% U2R 10.9% R2L 6.9%
2008	Xiang, Yong, & Meng	-Clusterización -NB -DT	Tasa de detección Normal 96.8% Probe 93.40% DOS 98.66% U2R 71.43%

Año	Autor(es)	Algoritmos utilizados	Resultados
			R2L 46.97%
2008	Giorgio Giacinto, Perdisci, Del Rio, & Roli	-Multiple Classifier System (MCS) -Parzen Classifiers -K means -v-SVC	Precisión 92.91%
2008	Weiming Hu, Wei Hu, & Maybank	AdaBoost	Tasa de detección 90.88% Tasa de Falsa alarma 0.31%
2008	Das, Schneider, & Neill	WSARE	Este estudio tomo 6 tipos comunes de ataques de KDD99, y se obtuvieron los siguientes tasas de detección apache2 96.68% guess password 77.92% mailbomb 22.43% Neptune 99.38% Smurf 76.62% Snmpguess 97.73%
2009	Tran, Cao, Tran, & Nguyen	Boosted Subspace Probabilistic Neural Network (BSPNN) (Adaptive Boosting y Semi-parametric NN)	Tasa de detección Normal 99.8% Probe 99.3% DOS 98.1% U2R 89.7% R2L 48.2%
2011	Farid, Rahman, & Rahman	-Boosting -NB -Adaboost	Tasa de detección Normal 100% Probe 99.95% DOS 99.92%

Año	Autor(es)	Algoritmos utilizados	Resultados
			U2R 99.55% R2L 99.60%
2011	Yi, Wu, & Xu	Improved incremental SVM algorithm (RS-ISVM)	Tasa de detección 89.175%
2011	Sangkatsanee, Wattanapongsakorn, & Charnsripinyo	-DT -Ripper Rule -Back-Propagation ANN	Tasa de detección Normal 99.979% Probe 98.868% DOS 99.434%
2012	Sivatha Sindhu, Geetha, & Kannan	DT	Porcentaje de detección 98.38% Porcentaje de error 1.62%
2012	S.-W. Lin, Ying, Lee, & Lee	-SVM -DT -Simulated Annealing (SA)	Precisión 99.96%
2013	Baig, Sait, & Shaheen	Group Method for Data Handling (GMDH)	Tasa de detección 97.72%
2014	KumarShrivas & Kumar Dewangan	ANN-Bayesian Net-GR (Redes Neuronales Artificiales, Redes bayesianas con Radio de ganancia GR)	Precisión KDD99 99.42% NSL KDD 98.07%
2015	Lin, Ke, & Tsai	Cluster center and nearest neighbor (CANN)	Precisión Normal 97.04% Probe 87.61% DOS 99.68% U2R 3.85% R2L 57.02%

Año	Autor(es)	Algoritmos utilizados	Resultados
2015	Srimuang & Intarasothonchun	Weighted Extreme Learning Machine	Precisión Probe 96.64% DOS 99.95% U2R 99.97% R2L 93.64%
2016	Aburomman & Ibne Reaz	-PSO -Local unimodal sampling (LUS) -Weighted majority algorithm (WMA)	Precisión PSO Normal 83.6121% Probe 96.6770% DOS 98.8388% U2R 99.7993% R2L 84.7715% LUS Normal 83.6878% Probe 96.8576% DOS 98.8534% U2R 99.8029% R2L 84.7615% WMA Normal 65.794% Probe 96.2464% DOS 98.4448% U2R 99.7939% R2L 82.897%
2017	Siddiqui & Farooqui	-SVM -Graph Technique ANN	Tasa de detección Normal 95.8% Probe 93.83%

Año	Autor(es)	Algoritmos utilizados	Resultados
			DOS 93.83% U2R 95.5% R2L 96.5% Tasa de precisión Normal 85.85% Probe 86.28% DOS 81.97% U2R 87.22% R2L 86.38% Recall Normal 84.84% Probe 83.8% DOS 80.97% U2R 84.05% R2L 84.85%
2017	Li et al.	Polynomial Feature Correlation	Tasa de detección 100% Tasa falsos positivos 0.13%
2018	Verma, Anwar, Khan, & Mane	-XGBoost -AdaBoost	Precisión 99.86%
2018	Ali, Al Mohammed, Ismail, & Zolkipli	-Fast Learning Network (FLN) -Particle Swarm Optimization (PSO)	Precisión Normal 0.9978 Probe 0.9077 DOS 0.9837 U2R 0.9363 R2L 0.6364
2018	Abdullah,	-RF	Precisión 99.78%

Año	Autor(es)	Algoritmos utilizados	Resultados
	Alshannaq, Balamash, & Almabdy	-C4.5 -Partial Decision List (PART) Ensemble classifier	

Tabla 7. Investigaciones aprendizaje de máquina (Ensemble classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos KDD-99.

2.1.4. Aprendizaje de máquina para detección de red utilizando el conjunto de datos de referencia CICIDS2017

CICIDS2017 generado por Sharafaldin, Habibi, & Ghorbani (2018), del Instituto Canadiense para la Ciberseguridad (CIC) y de la Universidad de "New Brunswick" (UNB).

Toma en consideración 11 criterios de ataque como son: *DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan y Botnet* (Sharafaldin et al., 2018).

Nace por la necesidad de crear un conjunto de datos de referencia actualizado que tenga una gran variedad de tráfico y diversidad de ataques[2].

A continuación se presentan las investigaciones más destacadas referentes al uso de algoritmos de aprendizaje de máquina en sistemas de detección de intrusos en los cuales se utilizó como conjunto de referencia el CICIDS2017. Al igual que en el punto anterior las investigaciones están divididas en tres categorías (*Single classifiers, Hybrid classifiers y Ensemble classifiers*),

Single Classifiers

Año	Autor(es)	Algoritmos utilizados
2018	Aksu, Üstebay, Aydin, & Atmaca	-KNN -SVM

[2] En el capítulo 3 se realiza un análisis detallado de este conjunto de datos.

Año	Autor(es)	Algoritmos utilizados
		-DT
2018	Dogukan Aksu & Ali Aydin	SVM Deep Learning
2018	Radford, Richardson, & Davis	RNN
2018	Bulavas	-PCA -DT
2018	Min et al.	Autoencoders
2019	Singh, Raiwani, & Panwar	- OneR (One Rule) -Reduced Error Pruning Tree (REPTree)
2019	Wankhede & Kshirsagar	-RF -ANN
2019	Roopak, Yun Tian, & Chambers	-ANN -LSTM -CNN
2019	Abdulrahman & Ibrahem	-RF -C5.0 -NB -SVM
2019	Park & Park	ANN
2019	Lopez, Mohan, & Nair	-Multiomial Logistic Regression -KNN -RF -NB -ANN -Dense Neural Networks
2019	Kaur & Singh	LSTM

Tabla 8. Investigaciones aprendizaje de máquina (Single classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos CICIDS2017.

Hybrid Classifiers

Año	Autor(es)	Algoritmos utilizados
2018	Ustebay, Turgut, & Aydin	Deep Multilayer Perceptron (DMLP) Deep Learning
2018	Zhu, Ye, Wang, & Xu	Attention-base Multi-Flow LSTM (AMF-LSTM)
2018	Ahmim, Maglaras, Ferrag, Derdour, & Janicke	-DT -Rules-based concepts
2019	Ahmim, Ferrag, Maglaras, Derdour, & Janicke	-J48 -Forest PA -RF -Rep Tree -Jrip -FURIA -Ridor -MLP -RBF -LibSVM -SMO -NaiveBayes

Tabla 9. Investigaciones aprendizaje de máquina (Hybrid classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos CICIDS2017.

Ensemble Classifiers

Año	Autor(es)	Algoritmos utilizados
2018	Vijayanand, Devaraj, & Kannapiran	GA based feature selection

Año	Autor(es)	Algoritmos utilizados
2018	Bansal & Kaur	XGBoost
2018	Azwar, Murtaz, Siddique, & Rehman	XGBoost
2018	Resende & Drummond	GA
2019	Abuzneid, Faezipour, Abdulhammed, Abu Mallouh, & Musafer	-Auto-Encoder (AE) -PCA
2019	Yulianto, Sukarno, & Suwastika	- AdaBoost -Synthetic Minority Oversampling Technique (SMOTE) -PCA -Ensemble Feature Selection (EFS)
2019	McKay, Pendleton, Britt, & Nakhavanit	-OneR -KNN -J48 -ANN -NB -RF
2019	Zhou & Cheng	-C4.5 -RF -Forest by Penalizing Attributes (Forest PA)

Tabla 10. Investigaciones aprendizaje de máquina (Ensemble classifiers) en sistemas de detección de intrusos utilizando en conjunto de datos CICIDS2017.

2.2.Comparativa de redes neuronales para analítica de datos

A continuación se presentan estudios en los cuales se realiza un análisis comparativo entre diversos algoritmos de aprendizaje de máquina en diversas aplicaciones de la analítica de datos.

Übeyli (2007), propone comparar la precisión de clasificación de diversos modelos de aprendizaje de maquina como son la MLPNN (red neuronal de perceptrón multicapa), CNN, PNN (red neuronal probabilística), RNN y SVM aplicados en los sistemas de diagnóstico automatizados para la detección del cáncer de mama. El conjunto de datos utilizado para este propósito fueron todos los registros en la base de datos de cáncer de mama de Wisconsin. Los resultados demostraron que la SVM logró un diagnóstico más preciso que los otros modelos evaluados. Entre los modelos CNN y RNN, este último obtuvo la mayor precisión de detección.

Deng & Platt (2014), propusieron diversos métodos lineales y log-lineales para el aprendizaje en conjunto en los modelos DNN, CNN y RNN, en aplicaciones de reconocimiento del habla. El estudio utilizó un conjunto de datos de entrenamiento de 462 muestras de audio. Los modelos fueron probados tanto de manera individual y de formando sistemas conjuntos entre ellos. Los resultados muestran que el sistema conjunto formado entre los tres modelos obtiene el mejor porcentaje de exactitud.

Tong, Gu, & Yu (2016), realizaron un estudio comparativo entre los modelos de *Deep Learning:* Redes neuronales totalmente conectadas (DNN), Redes neuronales recurrentes (RNN) y Redes neuronales convolucionales (CNN), aplicado a la detección de actividad de voz la cual forma parte de los sistemas de reconocimiento de voz automático. En la comparación experimental de los tres modelos se utilizaron tres conjuntos de datos de entrenamiento, cada uno compuesto por 7138 expresiones de 83 hablantes. El estudio demuestra que el modelo RNN (LSTM) es el más robusto respecto a los modelos DNN y CNN en diversas circunstancias.

Bae, Choi, & Kim (2016), proponen una arquitectura de red neuronal para la clasificación de la escena acústica. El modelo se compone de una capa CNN, RNN (LSTM) y DNN. La capa LSTM extrae información secuencial de las características de audio consecutivas, la capa CNN aprende la localidad espectro-temporal de las imágenes del espectrograma, y la capa DNN resume las salidas de las dos capas anteriores para aprovechar las características complementarias de la LSTM y la CNN al combinarlas y mejorar las tareas de clasificación. El estudio utilizo el conjunto de datos *TUT acoustic scenes 2016*. Los resultados demuestran la superioridad del modelo conjunto frente a cada modelo individual, así también podemos destacar que el modelos RNN (LSTM) presenta mayor exactitud de clasificación que el modelo CNN.

Sathyanarayana et al. (2016), plantearon un estudio que buscaba predecir la calidad del sueño basado en los datos de actividad física durante el tiempo de vigilia. El conjunto de datos utilizado fue recogido por los investigadores mediante sensores de actigrafía a 92 adolescentes durante 1 semana completa. Los métodos evaluados fueron MLP, CNN, RNN, LSTM, TB-LSTM. Los resultados del estudio mostraron que el modelo CNN presenta una mayor especificidad y sensibilidad en la predicción de la calidad del sueño.

Thomas, Maszczyk, Sinha, Kluge, & Dauwels (2017), realizaron un estudio comparativo entre los modelos de aprendizaje profundo CNN y RNN (LSTM), y algoritmos de clasificación tradicional (KNN, C4.5, SVM, AdaBoost y MLP), aplicados en el componente de clasificación de un sistema BCI (*Brain-computer interface*). El estudio utiliza el conjunto de datos SSVEP, el cual consta de 256 grabaciones SSVEP de 11 sujetos (8 hombres y 3 mujeres) con una frecuencia de muestreo de 250 Hz. Los resultados muestran la superioridad de los modelos de aprendizaje profundo frente a los algoritmos de clasificación tradicionales. Así también se valida que el modelos RNN (LSTM) presenta una mayor exactitud de predicción frente al modelo CNN.

Shao, Cai, Liu, & Lu (2017), en este estudio se evalúa diversos modelos de aprendizaje profundo como son DBN (*Deep belief networks*), SDAE (*Stacked Denoising Auto-Encoders*), CNN y RNN (LSTM), aplicado a la clasificación de imagen y video

obtenido a través de sensores RGB-D. Los modelos fueron evaluados en tres conjuntos de datos de reconocimiento de imágenes como son *2D&3D object*, *RGB-D object* y *NYU Depth v1 indoor scene segmentation*, y en dos conjuntos de datos de reconocimiento de video como son *Sheffield Kinect Gesture* (SKIG) y el *MSRDailyActivity3D*. Los resultados demuestran una superioridad del modelo CNN para la clasificación y predicción de imágenes, mientras que el modelo LSTM presenta superioridad en la clasificación y predicción en videos.

Jeon et al. (2017), este estudio investiga el uso de redes neuronales profundas para la detección de drones comerciales en entornos de la vida real mediante el análisis de los datos de sonido. Los investigadores grabaron los sonidos producidos por algunos de los drones comerciales más populares, para posteriormente aumentar el conjunto de datos con grabaciones de sonidos ambientales. Los modelos de redes neuronales utilizados fueron GMM (*Gaussian Mixture Model*), CNN y RNN. Los resultados muestran que el modelo RNN presenta el mejor desempeño de detección basado en el F-Score, detectando un dron con solo 240ms de entrada de audio.

Cakir, Parascandolo, Heittola, Huttunen, & Virtanen (2017), este estudio propone un modelo conjunto de CNN y RNN para la detección de eventos de sonidos. Se utilizaron 4 conjuntos de datos para la evaluación del modelo, uno de ellos fue generado a partir de la mezcla de eventos de sonidos aislados, y los tres restantes se grabaron entornos de la vida real. Los resultados obtenidos en la investigación muestran que el modelo conjunto presenta mayor precisión de detección que los modelos por separado. Pero se observa un mayor desempeño de detección del modelo CNN frente al RNN.

Yin, Kann, Yu, & Schütze (2017) realizan un estudio comparativo entre los modelos CNN, RNN y GRU (*Gated Recurrent Unit*) para tareas de procesamiento de lenguaje natural. Las tareas evaluadas fueron: *Sentiment Classification* (SentiC) mediante el conjunto de datos *Stanford Sentiment Treebank* (SST), *Relation Classification* (RC) en el conjunto de datos *SemEval 2010 task 8*, *Textual Entailment* (TE) en el conjunto de datos *Stanford Natural Language Inference* (SNLI), *Answer Selection* (AS) en el conjunto de

datos *WikiQA*, *Question Relation Match* (QRM) en el conjunto de datos *WebQSP*, *Path Query Answering* (PQA) y *Part-of-Speech Tagging* (POS) en el conjunto de datos WSJ. Los resultados mostraron un mejor desempeño de GRU en SentiC, RC, TE y PQA, CNN tuvo un mejor desempeño en AS y QRM, y LSTM solo obtuvo mayor desempeño en *POS Tagging*.

Guo, Liu, Bakker, Guo, & Lew (2018) proponen un marco de trabajo que combina una CNN y RNN para abordar la tarea de clasificación jerárquica de imágenes. La CNN tiene la capacidad de obtener características discriminativas para las imágenes de entrada, mientras que la RNN optimiza la clasificación de etiquetas gruesas (superclase) y finas (clase). El estudio utiliza dos conjuntos de datos de entrenamiento CIFAR-100 e *ImageNet 2012*. Los resultados del estudio de muestran que el modelo conjunto CNN-RNN, utilizan los datos de entrenamiento de superclase para mejorar la clasificación de las etiquetas de clase. Este modelo conjunto supera ampliamente a otros modelos de aprendizaje de maquina como SVM, CNN y RNN tratados de manera individual.

Lago, De Ridder, & De Schutter (2018), e cuatro modelos de aprendizaje profundo (DNN, LSTM, GRU y CNN) para la predicción de los precios. Y los comparan con otros 27 enfoques anteriores utilizados para el mismo fin. El conjunto de datos utilizado fueron los datos del mercado de Bélgica en el periodo comprendido entre el 01/01/2010 y el 31/11/2016. Los resultados del estudio demuestran que los modelos DNN, LSTM y GRU presentan una precisión de predicción significativamente mayor que todos los otros modelos probados.

Vinayakumar, Soman, & Poornachandran (2018), evalúan diversas arquitecturas de aprendizaje profundo específicamente la RNN, I-RNN, LSTM, CNN, CNN-LSTM, para la detección de URL´s maliciosas. El conjunto de datos utilizado fue creado a partir de diversas listas que proveen URL tanto malignas como benignas (Alexa, *DMOZ directory*, *MalwareURL*, *MalwareDomains* y *MalwareDomainList*). Los resultados muestran que el modelo LSTM muestra una ligera superioridad en la precisión de clasificación frente a los otros modelos evaluados.

2.2.1. Comparativa entre CNN y RNN

Como se mencionó en el capítulo anterior, el objetivo principal de la presente investigación es el estudio comparativo entre las Redes Neuronales Recurrentes y Redes Neuronales Convolucionales, para determinar cuál red ofrece un mayor desempeño en la detección de intrusos de red. Es por ello que a continuación se realizará un análisis de las investigaciones que han realizado un estudio comparativo entre ambos modelos en diversas aplicaciones de la analítica de datos.

Bertero & Fung (2016), proponen un estudio comparativo entre los modelos de aprendizaje profundo CNN y RNN para la predicción y detección del humor en diálogos. Para este estudio los investigadores tomaron los diálogos de la serie de televisión "*The Big Bang Theory*". El estudio utiliza una combinación de funciones de nivel de palabra y de audio. Los resultados de la investigación reflejan que el modelo CNN presenta mejores niveles de detección y predicción frente a RNN.

Amoh & Odame (2016) comparan los modelos CNN y RNN para la detección de la tos. El conjunto de datos para el estudio los investigadores crearon una base de datos de grabaciones de sonido pulmonar de 14 voluntarios sanos: siete hombres y siete mujeres. Las edades de los sujetos van desde los 21 a los 30 años, con etnias compuestas por africanos, asiáticos y caucásicos. Los resultados muestran que el modelo CNN produce una mejor especificidad, mientras que la RNN mejor sensibilidad de detección.

Huang, Chiang, & Li (2017) llevaron adelante un estudio para el pronóstico de tráfico de red móvil. Se estudiaron tres modelos de aprendizaje profundo: RNN, CNN 3D, y una combinación CNN-RNN. El conjunto de datos utilizado fue el publicado por la empresa Telecom Italia en el año 2015, el cual incluye registros de telecomunicaciones, clima, noticias, redes sociales y electricidad de la ciudad de Milán y la Provincia de Trentino durante noviembre y diciembre de 2013. La investigación se centró solamente en los registros de telecomunicaciones. Los experimentos indican que el modelo conjunto CNN-RNN muestra una mayor precisión de pronóstico del 70% al 80%.

Selvin, Vinayakumar, Gopalakrishnan, Menon, & Soman (2017) desarrollaron un estudio comparativo de los modelos CNN, RNN y LSTM, para la predicción del mercado de valores para la empresas de la lista NSE. El conjunto de datos utilizado constó con el precio de acciones por minuto de 1721 empresas que se cotizaron en la NSE en el período de julio de 2014 a junio de 2015. Los resultados de la investigación muestran que el modelo CNN presenta mayor capacidad para identificar cambios en las tendencias o fluctuaciones de los precios del mercado.

Wu, Jiang, Xu, Zhi, & Xu (2017) hicieron una comparación de los modelos CNN y RNN aplicado al reconocimiento de entidades nombradas clínicas (NER), una tarea del procesamiento de lenguaje natural que permite extraer conceptos a partir de textos clínicos. El estudio utilizó dos conjuntos de datos i2b2 2010 y MIMIC II. Los resultados del estudio mostraron un mayor desempeño del modelo RNN para la tarea del NER.

Lo Bosco & Di Gangi (2017) compararon los modelos CNN y RNN para la clasificación de secuencias de ADN. El conjunto de datos utilizado para este estudio fue el 16S rRNA, del cual se extrajeron 300 secuencias de ADN que fueron agrupadas en cinco categorías taxonómicas *Phylum*, *Class*, *Order*, *Family* y *Genus*. Los resultados del estudio mostraron la superioridad del modelo CNN en las tareas de clasificación de cuatro de las categorías (*Phylum, Class, Order* y *Family*), mientras que el modelo RNN obtuvo mayor exactitud en la clasificación de la categoría *Genus*.

Bellantonio et al. (2017) este estudio propone un modelo hibrido entre CNN y RNN, para el análisis de expresiones de dolor facial basado en la visión computarizada. El conjunto de datos utilizada para el entrenamiento y pruebas de este modelos fue el UNBC-*McMaster Shoulder Pain database*. La base de datos contiene secuencias de videos faciales de participantes que habían sufrido dolor de hombro y estaban realizando una serie de pruebas de rango de movimiento activas y pasivas en sus miembros afectados y no afectados en múltiples ocasiones. El estudio demostró buenos resultados en la detección de dolor. Se realizó además una comparativa entre los modelos CNN y RNN por separado sobre el mismo conjunto de datos. El modelo CNN se basó en la información de un solo

cuadro, mientras que el modelo LSTM tiene en cuenta las variaciones en las imágenes en el eje temporal. La LSTM mostró un mayor porcentaje de exactitud en la detección.

Con respecto a la predicción de ataques, Cui, Long, Min, Liu, & Li (2018) comparan entre los modelos CNN y RNN para la detección de intrusos de red. El conjunto de datos de referencia utilizado fue el ISCX2012, el cual contiene tráfico de red capturado en un lapso de siete días (tres legítimos y cuatro maliciosos), se incluyen cuatro clases de ataques (BFSSH, *Infiltrating*, HttpDoS, DDoS). Los resultados del estudio demuestran que el modelo CNN es mejor en la clasificación binaria (normal/ataque) del tráfico de red, mientras que el modelos LSTM en la detección multi-clase.

2.3.Discusión

Con la revisión de la literatura de los trabajos realizados por varios autores, y que se encuentran relacionados con los objetivos planteados en esta tesis, se puede concordar en la predominancia del conjunto de datos KDD99, como conjunto de datos de entrenamiento para diversos algoritmos de aprendizaje de máquina para la detección de intrusos.

Se pudo constatar que en los últimos dos años se han venido evaluando modelos para la detección de intrusos con diversos algoritmos de aprendizaje de máquina haciendo uso del conjunto de datos de referencia CICIDS2017. Pero no se encontró ningún estudio comparativo entre los modelos CNN y RNN que hagan uso del mencionado conjunto de datos para entrenamiento y pruebas. Solamente se encontró un estudio comparativo que hacía uso del conjunto de datos ISCX2012.

CAPÍTULO III

SOLUCIÓN ADOPTADA

3.1.Introducción

En la actualidad los IDS, centran el proceso de detección de anomalías generalmente en un solo aspecto, ya sea este a través de cálculos estadísticos del tráfico de red procesado, a través de la comparación con firmas o mediante algún algoritmo de aprendizaje automático (Hindy et al., 2018).

Como se observó en el capítulo anterior, los algoritmos de aprendizaje de máquina utilizados para detectar anomalías son diversos, sin embargo, la presente investigación se enfoca en la comparación de dos modelos de aprendizaje profundo para la detección de anomalías en el conjunto de datos “CICIDS2017”: redes neuronales convolucionales y redes neuronales recurrentes.

3.2.Descripción de la arquitectura

3.2.1. Análisis de solución

La solución propuesta, requiere cumplir las siguientes necesidades funcionales:

- Consolidar y pre procesar el conjunto de datos CICIDS2017. Los datos deben ser clasificados y etiquetados de la siguiente manera: Cuando corresponden a tráfico de red maligno con uno (1), y cuando corresponden a tráfico de red benigno con cero (0).
- Crear dos modelos de predicción, mediante el entrenamiento de dos redes neuronales (RNN y CNN), para el aprendizaje profundo de tráfico de red considero benigno o maligno.

- Realizar la validación de los modelos de predicción, y el análisis descriptivo mediante la utilización de una matriz de confusión.

3.2.2. Diseño de solución

A continuación, se describe el diseño de la solución adoptada, la cual tiene como objetivo la preparación del conjunto de datos, el desarrollo de la Red Neuronal Convolucional y Recurrente, y la validación de ambos modelos.

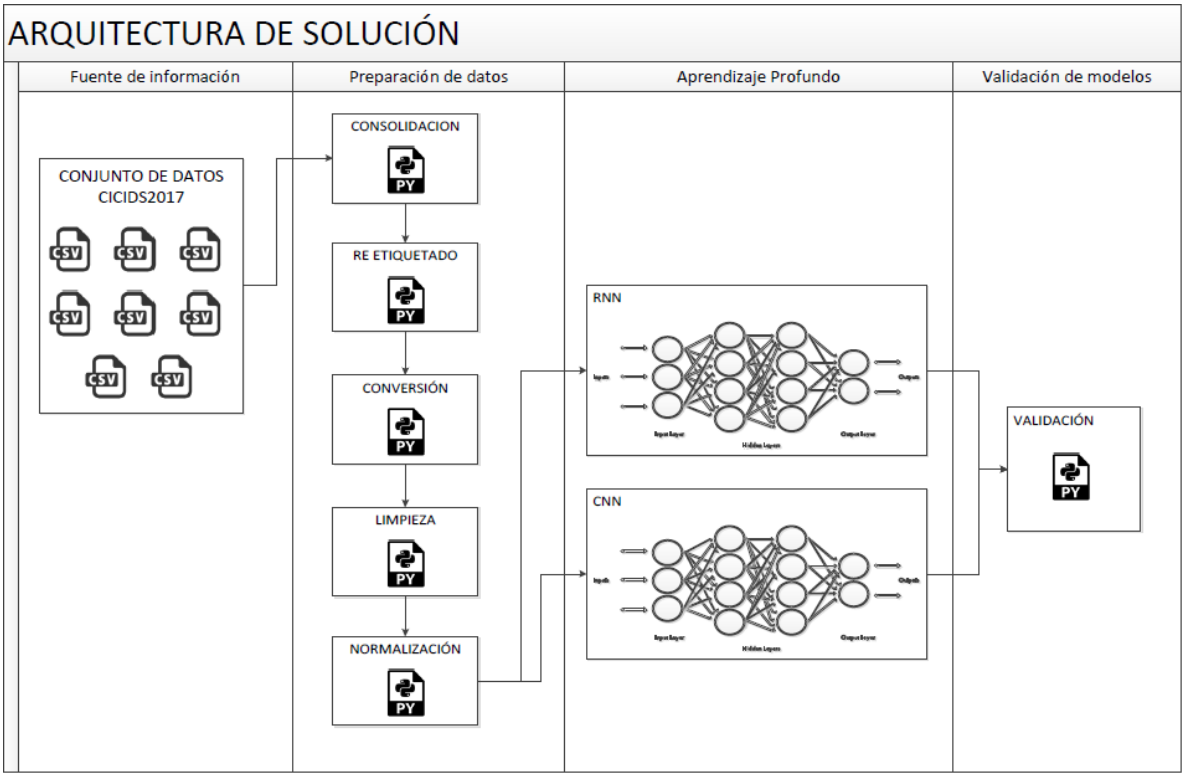

Figura 8. Arquitectura de la solución adoptada.

En la Figura 8, se puede observar la arquitectura estructurada en capas de la solución adoptada, permitiendo establecer un flujo, controlado e independiente de la información por capa.

En la primera capa se describe el conjunto de datos de referencia utilizado en la presente investigación. El conjunto de datos posteriormente es procesado en la segunda capa, siguiendo la siguiente secuencia

1. Se re etiquetan
2. Se validan y estandarizan los tipos de datos
3. Se eliminan los datos con formatos especiales, y,
4. Se normalizan.

A continuación, en la siguiente capa, se define la arquitectura que se utilizará para la creación de los modelos de predicción. Finalmente los modelos son validados mediante técnicas de validación cruzada para establecer su estabilidad y confianza.

3.2.2.1.Fuente de Información

En esta capa, se describe la fuente de la cual se obtuvo la información para entrenar ambos modelos. Para la selección de la fuente de información, se consideró el conjunto de datos CICIDS2017.

CICIDS2017 fue generado por Sharafaldin, Habibi, & Ghorbani (2018), del Instituto Canadiense para la Ciberseguridad (CIC) y de la Universidad de "*New Brunswick*" (UNB).

La principal razón que motivó a estos investigadores para la generación de este nuevo conjunto de datos fue, entre otros criterios, el hecho de que muchos de los conjuntos de datos de referencia utilizados en investigaciones sobre sistemas de detección de intrusos de red, no han podido ser abiertamente compartidos debido a asuntos relacionados con la privacidad de la información. Mientras, que los que sí han podido ser expuestos al público, no se han mantenido actualizados, reflejando una falta de variedad de tráfico y diversidad de ataques. Si a esto se agrega la evolución de los malwares y los cambios continuos en las técnicas de ataques, ha sido necesario que los conjuntos de datos de referencia para este tipo de investigaciones se encuentren en constante actualización (Sharafaldin et al., 2018).

Por lo antes mencionado CICIDS2017, fue generado tomando en consideración once criterios de ataque como son: *DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan y Botnet* (Sharafaldin et al., 2018). En contraposición con el conjunto de datos de referencia más utilizado en los últimos diez años según el estudio realizado por Hindy et al. (2018), el KDD-99 y su versión mejorada denominada NSL-KDD, que cuentan con solamente cuatro criterios de ataque.

Para la creación del mencionado conjunto de datos, los investigadores implementaron dos redes completamente separadas tanto a nivel lógico y de infraestructura, una red víctima y una red ataque. La red víctima fue constituida por un firewall, router, switch de acceso, servidores tanto Windows 2016 como Ubuntu 16 y agentes con los sistemas operativos: Ubuntu 14.4 y 16.4 (32 y 64bis), Windows 7 Pro, 8.1-64, Vista, 10 (Pro 32 y 64 bits) y Mac. Mientras que la red atacante fue constituida por un router, un switch de acceso y agentes con sistemas operativos: Kali y Windows 8.1 con direcciones IP públicas.

El direccionamiento IP utilizado en la implementación antes mencionada se puede observar en la siguiente tabla:

	Máquina	**Sistema Operativo**	**IPs**
Red Víctima	Servidores	Win Server 2016 (DC y DNS)	192.168.10.3
		Ubuntu 16 (Web Server)	192.168.10.50 – 205.174.165.68
		Ubuntu 12	192.168.10.51 – 205.174.165.66
	PCs	Ubuntu 14.4 (32,64 bits)	192.168.10.19-192.168.10.17
		Ubuntu 16.4 (32,64 bits)	192.168.10.16-192.168.10.12
		Win7 PRO	192.168.10.9
		Win8.1 PRO (64 bits)	192.168.10.5
		Win Vista	192.168.10.8

	Máquina	Sistema Operativo	IPs
		Win10 PRO (32-64 bits)	192.168.10.14-192.168.10.15
		Mac	192.168.10.25
	Firewall	Fortinet	
Red Atacante	PCs	Kali	205.174.165.73
		Win 8.1	205.174.165.69
		Win 8.1	205.174.165.70
		Win 8.1	205.174.165.71

Tabla 11. Sistemas operativos y direcciones IP de red víctima y atacante (Sharafaldin et al., 2018)

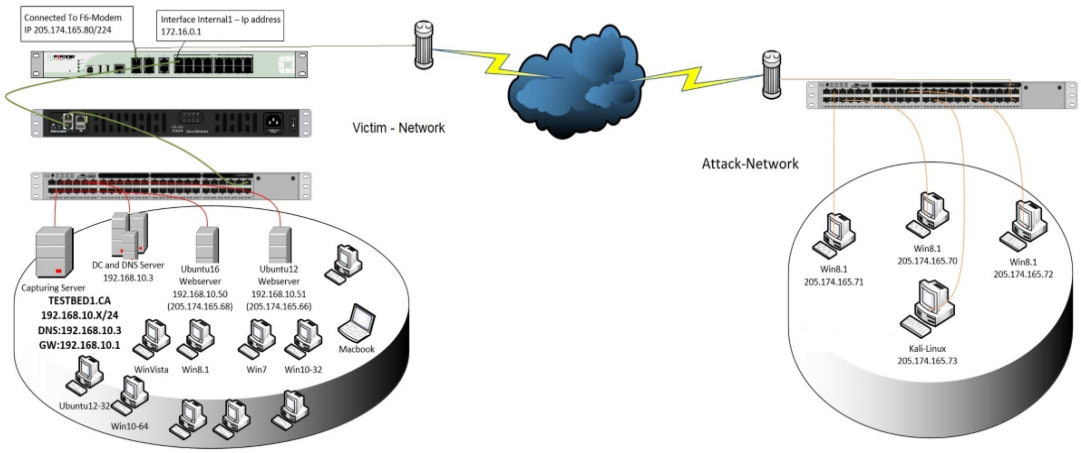

Figura 9. Diagrama de red CICIDS2017 (Sharafaldin et al., 2018).

El conjunto de datos fue creado en base a 6 perfiles generales de ataque:

- *Brute Force Attack* (Ataque de fuerza bruta)
- *Heartbleed Attack*
- *Botnet-*
- *DoS Attack*
- *DDoS Attack*

- *Web Attack*, dentro de este perfil CICIDS2017 incluye las siguientes técnicas de ataque: *SQL Injection*, *Cross-Site Scripting* (XSS) y *Brute Force over HTTP (Ataque de fuerza bruta sobre HTTP)*.
- *Infiltration Attack*.

El lapso de tiempo sobre el cual fue capturado el tráfico de red, fue entre el lunes 3 de julio de 2017 a las 09:00 y el viernes 7 de julio a las 17:00. Cada perfil de ataque fue ejecutado consecuentemente uno tras otro durante este periodo de la siguiente manera:

Días	Etiquetas
Lunes	Benign
Martes	BForce, SFTP y SSH
Miércoles	DoS y Heartbleed Attacks: slowloris, SLowhttptest, Hulk y GoldenEye
Jueves	Web e Infiltration attacks: Web BForce, XSS, SQL Inject, Infiltration Dropbox Download y Cool disk
Viernes	DDoS LOIT, Botnet ARES PortScans (sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL, B)

Tabla 12. Etiquetas de ataque diario CICIDS2017 (Sharafaldin et al., 2018)

Para la ejecución de cada ataque se utilizaron, a criterio de los investigadores, las mejores herramientas desarrolladas en *Python* que se hayan encontrado públicamente, de la siguiente manera:

- *Bruteforce Attack*, se utilizó una maquina con sistema operativo *Kali Linux*[3] ejecutando la herramienta *FTP*y *SSH Patator*[4] contra el servidor Web Ubuntu 16.04.
- *DoS Attack*, mediante la herramienta *Heartleech*[5] se explota la vulnerabilidad de *OpenSSL* v1.0 instalada en el servidor web Ubuntu 12.04, obteniendo volcados de memoria del servidor.

[3] https://www.kali.org/
[4] https://github.com/lanjelot/patator
[5] https://github.com/robertdavidgraham/heartleech

- *Web Attack,* para la ejecución de los ataques *SQL Injection*, *XSS* y *Brute Force over HTTP*, los investigadores desarrollaron su propias herramientas, las cuales fueron ejecutadas en una máquina con sistema operativo Kali Linux. Estos ataques fueron ejecutados sobre el servidor web Ubuntu16.04, en el cual instalaron la aplicación web DVWA[6] (*Damn Vulnerable Web App*), una aplicación *PHP/MySQL*, diseñada por el *DVWA team*, específicamente para ser vulnerable y poder ser explotada con fines educativos.
- *Inflitration Attack*, para este escenario de ataque se utilizó *Metasploit*[7], el cual generó un archivo infectado. Este fue descargado desde Dropbox en las máquinas Windows y colocado en la máquina Mac mediante un disco externo. El archivo infectado ejecutó las herramientas *Nmap* y *portsan* en toda la red víctima.
- *Botnet Attack*, para este ataque se utilizó *Ares*[8], un *botnet* escrito en *Python* que permite conexiones remotas, subida y bajada de archivos, entre otros. La máquina atacante contaba con el sistema operativo Kali Linux, y el *botnet* fue ejecutado en 5 diferentes maquinas con Windows: Vista, 7, 8.1, 10 (32 y 64 bits).
- *DDoS Attack* , para el ataque *DDoS* se utilizó LOIC[9] (*Low Orbit Ion Canon*), el cual envió tráfico UDP, TCP y solicitudes HTTP al servidor web Ubuntu 16. Las máquinas Windows 8.1 ejecutaron la herramienta contra el servidor web Ubuntu 16.
- *PortScan*, para este ataque se ejecutó Nmap[10] sobre todas las máquinas Windows, cambiando las banderas de Nmap sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL y B.

Todo el tráfico de red capturado tanto benigno como maligno, fue analizado mediante la herramienta *CICFlowMeter*[11], con la cual permitió extraer las 79 características que definen a cada flujo de datos. Entre las características más representativas se tienen dirección IP de origen, dirección IP de destino, puerto de origen, puerto de destino,

[6] http://www.dvwa.co.uk/
[7] https://www.metasploit.com/
[8] https://github.com/sweetsoftware/Ares
[9] https://github.com/NewEraCracker/LOIC
[10] https://nmap.org/
[11] http://www.netflowmeter.ca/

protocolo, duración del flujo, total de paquetes del flujo, entre otras. La descripción de las 79 características que se pueden extraer con *CICFlowMeter* se puede encontrar en la página web de la herramienta[12].

3.2.2.2.Preparación de los datos

En esta capa, se definieron los procesos preparatorios que se usaron para mejorar la calidad del conjunto de datos.

Consolidación

El conjunto de datos CICIDS2017 está formado por 9 archivos .csv, a los cuales se los consolidó en un solo archivo .csv con todo el tráfico de red, mediante el siguiente algoritmo:

1. Inicio
2. Se define la ruta *"./MachineLearningCVE/"* donde se encuentran los 9 archivos .csv que conforman el conjunto de datos CICIDS2017
3. Se define un arreglo vacío llamdo *dfs*
4. Se define un *dataframe* vacío llamado *datos*
5. Por cada archivo .csv encontrado en la ruta:
    i. Lee el archivo .csv
    ii. Adjunta los datos leídos del .csv en *dfs* de forma secuencial
6. Se copia todos los datos de *dfs* a *datos*
7. Se guarda los datos como un nuevo archivo .csv llamado “cicids2017.csv”
8. Fin

[12] http://www.netflowmeter.ca/netflowmeter.html

Este algoritmo lee todos los datos de los archivos .csv en una ruta determinada, copia uno a uno los datos de cada .csv en un *dataframe*, para finalmente guardar todos los datos copiados en un nuevo archivo .csv denominado "*cicids2017.csv*".

Este proceso permitió trabajar con todos los datos en simultáneo, considerando que para la presente investigación no es de interés el día en que fue realizado el ataque, ni el tipo de ataque, solo conocer si el tráfico de red es benigno o maligno.

Re etiquetado

Como se mencionó, el objetivo de la investigación es determinar la detección de intrusos de red, para esto los modelos RNN y CNN deberán ser capaces de clasificar entre tráfico benigno y maligno.

Inicialmente el conjunto de datos tiene etiquetado el tráfico benigno o normal con la palabra *Benign*, y el tráfico generado por un ataque específico (maligno) se encuentra etiquetado con el nombre del ataque específico:

- *DoS GoldenEye*
- *Heartbleed*
- *DoS Hulk*
- *DoS Slowhttp*
- *DoS slowloris*
- *SSH-Patator*
- *FTP-Patator*
- *Web Attack*
- *Infiltration*

- *Bot*
- *PortScan*
- *DDoS*

Como el objetivo de los modelos RNN y CNN es la clasificación entre tráfico benigno y maligno, se procedió a re etiquetar el conjunto de datos consolidado de la siguiente manera, trafico benigno con la etiqueta 0 y tráfico maligno con la etiqueta 1. Para lo cual se utilizó el siguiente algoritmo:

1. Inicio
2. Se define la ruta *"./MachineLearningCVE/"* donde se encuentran el archivo cicids2017.csv
3. Se lee el archivo "*cicids2017.csv*" en un *dataframe*
4. Se transforma el *dataframe* a una lista
5. Por cada elemento de lista:
    i. Si la columna 78 del elemento es igual a "*BENING*":

        Se coloca en la columna 78 del elemento un "0"

    ii. Caso contrario:

        Se coloca en la columna 78 del elemento un "1"
6. Se guarda la lista como un nuevo archivo .csv llamado "*cicids2017_label1.csv*"
7. Fin

El algoritmo lee todo el conjunto de datos consolidado, posteriormente en cada línea que conforma el conjunto de datos, accede a la etiqueta ubicada en la columna número

78, y realiza el siguiente cambio: si encuentra la palabra *benign* coloca un 0, caso contrario coloca un 1. Una vez realizada esta acción guarda los datos re etiquetados en un .csv denominado "*cicids2017_label1.csv*".

Conversión de tipo de datos

Con el conjunto de datos re etiquetado, se validó el tipo de dato en cada columna, esto con el objetivo de verificar que cada una de ellas tenga un tipo de dato numérico, ya sea este flotante o entero. La validación se realizó con el siguiente algoritmo:

1. Inicio
2. Se define la ruta *"./MachineLearningCVE/"* donde se encuentran el archivo "*cicids2017_label1.csv*"
3. Se lee el archivo "*cicids2017_label1.csv*" en un *dataframe*
4. Se imprime los tipos de datos de cada columna del *dataframe*
5. Se convierte las columnas *'Flow Bytes/s'* y ' *Flow Packets/s*' del *dataframe* a *float* (flotantes)
6. Se guarda el *dataframe* como un nuevo archivo .csv llamado "*cicids2017_label2.csv*"
7. Fin

Se verificó con este algoritmo que las columnas *Flow Bytes/s* y *Flow Packets/s del archivo* "*cicids2017_label1.csv*" poseían un tipo de dato *object.* Este tipo de dato no coincide con los datos contenidos en las columnas, por lo que se cambió el tipo de dato a flotante, y se guardó todos los datos en un nuevo archivo .csv con el nombre "*cicids2017_label2.csv*".

Limpieza de datos

Luego de haber convertido los datos, se validó si en estos existen valores especiales que puedan introducir errores en el entrenamiento de las redes neuronales, para esto se verificó la existencia de dos tipos valores especiales: valores nulos (*NaN*) y valores *inf*. Con este procedimiento de limpieza se redujo un 0,1% del total del conjunto de datos, debido a que poseían datos con estos valores especiales.

Mediante el siguiente algoritmo se detectó y eliminó 1358 filas con valores nulos (*NaN*):

1. Inicio
2. Se define la ruta *"./MachineLearningCVE/"* donde se encuentran el archivo "*cicids2017_label2.csv*"
3. Se lee el archivo "*cicids2017_label2.csv*" en un *dataframe*
4. La función *dropna()*, busca y elimina los todas las filas que contengan el valor *NaN*
5. Se guarda el *dataframe* sin los valores *NaN* como un nuevo archivo .csv llamado "*cicids2017_label3.csv*"
6. Fin

Además de los valores nulos, se eliminaron 1509 filas que contenían valores *inf* mediante el siguiente algoritmo:

1. Inicio
2. Se define la ruta *"./MachineLearningCVE/"* donde se encuentran el archivo "*cicids2017_label3.csv*"
3. Se lee el archivo "*cicids2017_label3.csv*" en un *dataframe*
4. Mediante la función *isin()*, se busca los valores *inf*

5. Se obtiene los índices de las filas con los valores *inf*

6. Se elimina las filas con los índices obtenidos en el paso anterior

7. Se guarda el *dataframe* sin los valores *inf* como un nuevo archivo .csv llamado "*cicids2017_label4.csv*"

8. Fin

Normalización de datos

La normalización es un proceso de transformación que permite redefinir el conjunto de datos para ajustar los valores iniciales en diferentes escalas a una escala común (Hernández & Rodríguez, 2008).

El proceso de normalización ejecuta una transformación lineal de los datos originales. Con base en los valores mínimo y máximo de los datos, se calcula un valor de normalización *v'* en base al valor *v*, de acuerdo a la siguiente expresión (Hernández & Rodríguez, 2008):

$$v' = \frac{v - min_A}{max_A - min_A}\left(nuevo_{max_A} - nuevo_{min_A}\right) + nuevo_{min_A}$$

Ejemplo: suponiendo que el valor máximo y mínimo de un conjunto de datos son 15 y 85 respectivamente, y se requiere mapear los valores en un rango entre 0 y 1. Entonces, tomando un valor de 50 por normalización es transformado en:

$$v' = \frac{50 - 15}{85 - 15}(1 - 0) + 0 = 0{,}5$$

Mediante el procedimiento antes descrito se normalizaron todos los valores del conjunto de datos entre 0 y 1, considerando que las salidas esperadas de los modelos RNN y CNN son 0 y 1, para lo cual se ejecutó el siguiente algoritmo:

1. Inicio

2. Se define la ruta *"./MachineLearningCVE/"* donde se encuentran el archivo "*cicids2017_label4.csv*"

3. Se lee el archivo "*cicids2017_label4.csv"* en un *dataframe*

4. Se re escala los valores entre 0 y 1

5. Se guarda el *dataframe* con los valores re escalados como un nuevo archivo .csv llamado "*cicids2017_label5.csv*"

6. Fin

3.2.2.3.Aprendizaje Profundo

En esta capa, se definen las arquitecturas de la red neuronal convolucional (CNN) y de la red neuronal recurrente (RNN), que se utilizaran para la obtención de los modelos de predicción.

Para la creación de los modelos CNN y RNN, se utilizó programas desarrollados en *Python*, con un módulo de inteligencia artificial denominado *Keras*[13].

Arquitectura CNN

Como se describió en el punto 1.4.5.3 del capítulo I, la arquitectura de una CNN es:

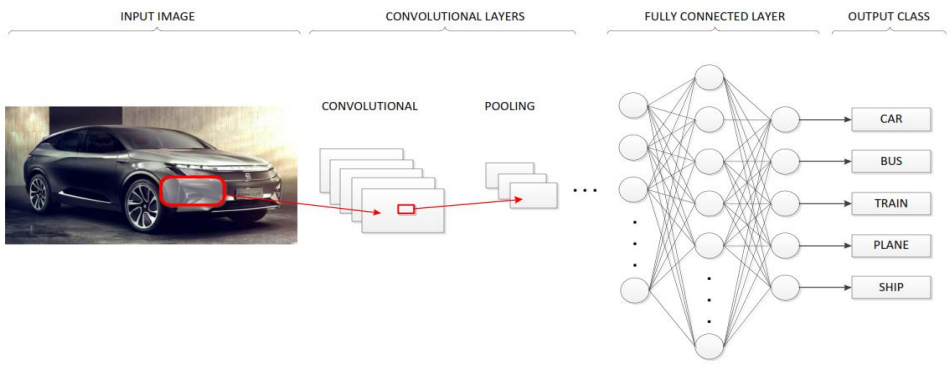

Figura 10. Arquitectura CNN (Zaragoza, 2018)

[13] https://keras.io/

Para este modelo se utilizó una capa convolucional 1D, debido a que esta capa es muy efectiva para obtener características de segmentos de datos de longitud fija dentro de un conjunto de datos general, donde la ubicación de la característica dentro del sistema no es relevante(Ackermann, 2018).

Los parámetros que se configuraron en la capa convolucional fueron (MkDocs, 2014):

- *Filters*, que constituyen los filtros que se utilizan para el aprendizaje de las características de una capa a otra. En este caso se colocaron 78 filtros ya que el conjunto de datos posee 78 características.
- *Kernel_size*, representa el tamaño de la ventana convolución. En este caso se utilizó un tamaño de kernel de 77, lo que permite obtener setenta y siete vectores de características por cada filtro.
- *Activation*, representa la función de activación que se usa en las capas de convolución, en este caso se utilizó la función "*relu*", para aprendizaje de límites de decisión no lineales.

Se utilizó además una capa de *Maxpooling* que permitió reducir la complejidad de la salida de la capa convolucional, evitando el sobre ajuste de la red. Para este modelo se utilizó un tamaño de ventana de 2, lo que redujo a la mitad la salida de la capa convolucional. Un tamaño de ventana mayor podría afectar a la red, ya que posiblemente se pierdan ciertas características que fueron detectadas por la capa convolucional. (Ackermann, 2018).

Se colocó también capas de *Dropout* que ayudaron a prevenir el sobre entrenamiento y permitieron combinar las diferentes capas en la arquitectura de la red neuronal (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

Finalmente la CNN propuesta culmina con una capa completamente conectada, que sirve de clasificador en base a las características obtenidas en las capas anteriores. Para la elección del número de sub capas y neuronas, que confirman esta capa totalmente conectada, se utilizó un método experimental donde se plantearon varias configuraciones

para la capa totalmente conectada, y se observó el comportamiento de la exactitud, pérdida y el tiempo de entrenamiento de toda la CNN en cada configuración.

A continuación, se presentan las configuraciones probadas de cada CNN y los resultados obtenidos:

Capa Convolucional	Capa Pooling	Capa Totalmente Conectada				Capa de Salida	#Épocas	Resultados	
		# Sub Capas	# Neuronas en cada Sub Capa			#Neuronas		Precisión	Tiempo de Entrenamiento (min)
1D	Maxpolling Ventana 2	2	156	78		1	5	0,95708	73,70
			156	39				0,95857	65,53
			156	26				0,95267	64,80
			156	13				0,95553	65,67
			78	39				0,95561	40,25
			78	26				0,95428	40,40
			78	13				0,95476	39,72
			39	26				0,94992	30,38
			39	13				0,93476	31,40
			26	13				0,94758	27,27
			156	78			10	0,96044	124,42
			156	39				0,96401	122,98
			156	26				0,96041	123,88
			156	13				0,96246	123,87
			78	39				0,96273	75,32
			78	26				0,96074	75,32
			78	13				0,95890	75,48
			39	26				0,95752	57,10
			39	13				0,95496	57,11
			26	13				0,94661	51,93
			156	78			15	0,96469	189,87
			156	39				0,96853	188,67
			156	26				0,96502	187,78
			156	13				0,96705	188,85
			78	39				0,96014	112,08
			78	26				0,96069	113,55
			78	13				0,96230	112,83

Capa Convolucional	Capa Pooling	Capa Totalmente Conectada				Capa de Salida	#Épocas	Resultados	
		# Sub Capas	# Neuronas en cada Sub Capa			#Neuronas		Precisión	Tiempo de Entrenamiento (min)
			39	26				0,96109	85,93
			39	13				0,95053	86,20
			26	13				0,95519	77,73
		3	156	78	39		5	0,95703	67,40
			156	78	26			0,94187	67,22
			156	78	13			0,95321	68,52
			156	39	26			0,94680	63,12
			156	39	13			0,95193	59,87
			156	26	13			0,95390	61,05
			78	39	26			0,94330	33,60
			78	39	13			0,94316	32,58
			78	26	13			0,94263	32,43
			39	26	13			0,91863	28,13
			156	78	39		10	0,95774	110,22
			156	78	26			0,95866	110,13
			156	78	13			0,95756	110,03
			156	39	26			0,95588	110,28
			156	39	13			0,95373	105,70
			156	26	13			0,95872	107,92
			78	39	26			0,93424	63,65
			78	39	13			0,95843	63,55
			78	26	13			0,94126	80,73
			39	26	13			0,95130	61,47
			156	78	39		15	0,95670	173,77
			156	78	26			0,96123	172,02
			156	78	13			0,95639	171,10
			156	39	26			0,96156	175,57
			156	39	13			0,96204	178,00
			156	26	13			0,95123	176,68
			78	39	26			0,95809	97,78
			78	39	13			0,95358	98,93
			78	26	13			0,95508	98,55
			39	26	13			0,95556	88,02

Tabla 13. Arquitecturas CNN.

En función de los resultados obtenidos, se escogió la arquitectura con la exactitud promedio entre todos los experimentos realizados. Como se puede observar en la Tabla 13 (fila subrayada) esta arquitectura fue la conformada por: la capa convolucional 1D, la capa de *Maxpooling*, la capa totalmente conectada formada por 2 sub capas con 78 y 13 neuronas respectivamente y la capa de salida con una neurona. Esta red tiene una exactitud del 95,476% y un tiempo de entrenamiento de 39,72 minutos, por lo que se escogió esta red para obtener el modelo de predicción para la CNN

Arquitectura RNN

Como se puede ver en la figura 11, la arquitectura de una RNN está conformada por una capa de entrada, un conjunto de capas ocultas y una capa de salida. A su vez cada capa está conformada por unidades de memoria a corto plazo largo (*LSTM units*):

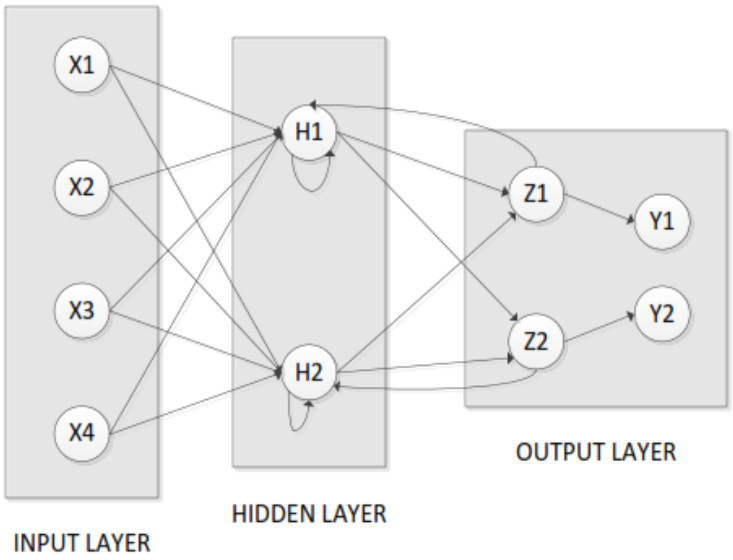

Figura 11. Arquitectura RNN (Khosla, Ramesh, Sharma, & Nyakotey, 2018)

Para determinar el número de unidades LSTM que se utilizaron en cada capa, así como el número de capas ocultas, para determinar la arquitectura de la RNN igualmente se lo realizó mediante un método experimental. Para esto, se plantearon distintas arquitecturas

en las cuales se variaron el número de capas ocultas y el número de unidades LSTM en cada capa, y se observó el comportamiento de la red en cuanto a la exactitud, pérdida y el tiempo de entrenamiento de la red.

A continuación se presentan las configuraciones probadas de cada RNN y los resultados obtenidos:

Capa Entrada	**Capa Oculta**				**Capa de Salida**	**# Épocas**	**Resultados**	
# Unidades LSTM	**# de Capas Ocultas**	**# Unidades LSTM en cada Capa Oculta**			**# Unidades LSTM**		**Precisión**	**Tiempo de Entrenamiento (min)**
78	2	156	78		1	5	0,97191	34,50
		156	39				0,97401	30,78
		156	26				0,97010	29,98
		156	13				0,97323	29,40
		78	39				0,97117	21,42
		78	26				0,97469	21,62
		78	13				0,97182	23,28
		39	26				0,97245	21,90
		39	13				0,97149	23,18
		26	13				0,97096	25,01
		156	78			10	0,97489	68,22
		156	39				0,97406	65,98
		156	26				0,96997	60,10
		156	13				0,97659	59,17
		78	39				0,97675	45,87
		78	26				0,97447	45,67
		78	13				0,97505	44,87
		39	26				0,97126	42,77
		39	13				0,97506	41,27
		26	13				0,97376	42,47
		156	78			15	0,97344	94,42
		156	39				0,97657	86,40

Capa Entrada	Capa Oculta				Capa de Salida	# Épocas	Resultados	
# Unidades LSTM	# de Capas Ocultas	# Unidades LSTM en cada Capa Oculta			# Unidades LSTM		Precisión	Tiempo de Entrenamiento (min)
		156	26				0,97505	86,07
		156	13				0,97033	88,20
		78	39				0,97614	64,00
		78	26				0,97334	65,70
		78	13				0,97677	74,03
		39	26				0,97716	66,33
		39	13				0,97220	69,55
		26	13				0,97041	73,60
		156	78	39			0,97120	36,57
		156	78	26			0,97274	37,33
		156	78	13			0,96922	39,12
		156	39	26			0,97288	33,87
		156	39	13			0,97414	33,98
		156	26	13		5	0,96873	33,80
		78	39	26			0,97394	25,60
		78	39	13			0,96999	27,37
		78	26	13			0,97030	30,17
		39	26	13			0,96979	27,40
	3	156	78	39			0,97579	70,93
		156	78	26			0,97607	71,77
		156	78	13			0,97308	75,05
		156	39	26			0,97666	73,17
		156	39	13			0,97675	79,48
		156	26	13		10	0,97517	82,15
		78	39	26			0,97101	69,15
		78	39	13			0,97008	76,60
		78	26	13			0,97499	84,03
		39	26	13			0,97611	68,08
		156	78	39		15	0,97720	104,25

Capa Entrada	Capa Oculta				Capa de Salida	# Épocas	Resultados	
# Unidades LSTM	# de Capas Ocultas	# Unidades LSTM en cada Capa Oculta			# Unidades LSTM		Precisión	Tiempo de Entrenamiento (min)
		156	78	26			0,97720	105,55
		156	78	13			0,97318	115,17
		156	39	26			0,97653	104,88
		156	39	13			0,97647	111,18
		156	26	13			0,97458	106,03
		78	39	26			0,97149	93,53
		78	39	13			0,97317	96,58
		78	26	13			0,96949	100,05
		39	26	13			0,97680	83,60

Tabla 14. Arquitecturas RNN

En función de los resultados obtenidos, se escogió la arquitectura con la exactitud promedio entre todos los experimentos realizados. Como se puede observar en la Tabla 14 (fila subrayada) esta arquitectura fue la conformada por la capa de entrada con 78 unidades LSTM, 2 capas ocultas con 156 y 13 unidades LSTM respectivamente y la capa de salida con una unidad. Esta red tiene una exactitud del 97,323% y un tiempo de entrenamiento de 29,40 minutos., por lo que se escogió esta red para obtener el modelo de predicción para la RNN.

3.2.2.4.Validación de modelos

Una vez seleccionado lo modelo para cada tipo de red neuronal, se realizó la validación de estos, para lo cual se utilizaron técnicas de validación cruzada detalladas en el capítulo 4.

CAPÍTULO IV

VALIDACIÓN

4.1. Método

En el capítulo anterior, se explicó en detalle cómo se obtuvieron las arquitecturas de la Red Neuronal Convolucional (CNN) y de la Red Neuronal Recurrente (RNN). A partir de estas arquitecturas se realizó el entrenamiento de las mismas, para obtener los modelos de predicción respectivos, de acuerdo al siguiente diagrama de flujo:

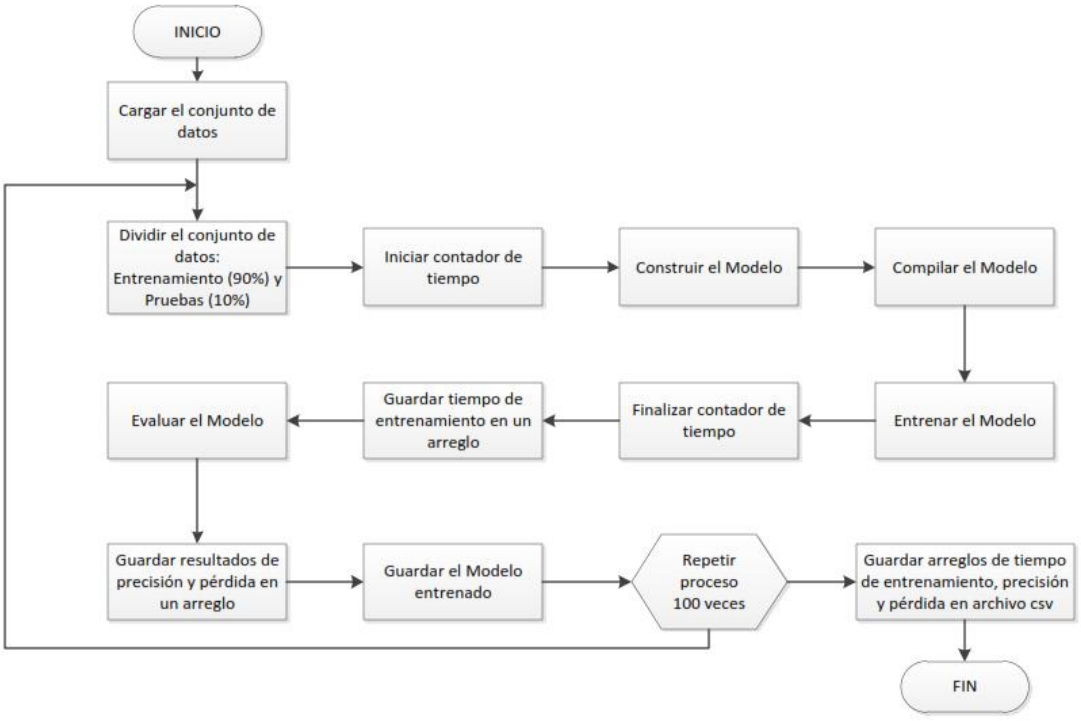

Figura 12. Diagrama de flujo de entrenamiento RNN y CNN

Como se puede apreciar en la figura 12, inicialmente se cargó el conjunto de datos "*cicids2017_label5.csv"* en un *dataframe*. Este fue convertido en una lista, la cual fue

separada en dos listas diferentes, la primera lista incluía la información de los flujos de datos, y la segunda lista su respectiva etiqueta (0 o 1).

A continuación mediante la función *StratifiedShuffleSplit,* se dividió el conjunto de datos contenido de las dos listas antes mencionadas, en dos subconjuntos: uno de entrenamiento y otro de pruebas. Esta división se realizó de manera aleatoria, preservando un porcentaje de muestras para cada clase. La proporción utilizada fue de 90% para datos de entrenamiento y 10% para datos de pruebas como se observa en la tabla 15.

	% Division	**Flujos de Red**
Datos de entrenamiento	90%	2’545,088
Datos de prueba	10%	282,788
Total	**100%**	**2827876**

Tabla 15. División de datos para entrenamiento y pruebas

La función *StratifiedShuffleSplit,* tiene la particularidad de realizar el proceso de división un número de veces determinado, en este caso realizó el proceso de división 100 veces, uno por cada entrenamiento de la red neuronal.

En cada iteración de la función *StratifiedShuffleSplit*, se realizaron los siguientes subprocesos:

1. Se inicializó un contador de tiempo para conocer el tiempo que duró cada entrenamiento.

2. Se construyó el modelo en base a las dos arquitecturas que fueron validadas en el capítulo 3 para cada una de las redes neuronales.

3. Una vez construido el modelo, se compiló el mismo, mediante la librería *Keras*.

4. Con el modelo compilado se entrenó el mismo, mediante la función *fit,* a la cual se le pasó el subconjunto de datos de entrenamiento que fue generado en pasos anteriores.

5. Cuando el entrenamiento culminó, se finalizó el contador de tiempo, y se guardó el tiempo de entrenamiento transcurrido en un arreglo.

6. El modelo entrenado fue evaluado mediante la función *evaluate*, utilizando para esto el subconjunto de datos de pruebas generado en pasos anteriores. La función *evaluate* obtuvo los valores de precisión y pérdida del modelo entrenado. Y se guardó estos resultados en un arreglo.

7. El modelo entrenado se guardó en un archivo tipo *.h5*.

Una vez se ejecutaron las 100 iteraciones, finalmente los arreglos con los datos de la precisión, perdida y el tiempo de entrenamiento fueron consolidados en un solo archivo *.csv*.

4.2.Resultados

Los resultados obtenidos en las validaciones de los modelos entrenados en el punto anterior son presentados a continuación en forma de gráficos de puntos. Para esto, se presentan tres gráficas (precisión, pérdida y tiempo de entrenamiento) para cada red neuronal. Mediante estos gráficos se busca verificar la estabilidad de los resultados de precisión, pérdida y tiempo de entrenamiento de los modelos generados, para lo cual en cada gráfico se dibujó la línea de tendencia en base a los resultados obtenidos de cada red neuronal.

Los ejes de las figuras 13, 14, 15, 16, 17 y 18 están definidos por:

- Eje x, está representado por el número de entrenamientos de la red neuronal.
- Eje y, está representado por las medidas de precisión, perdida y tiempo de entrenamiento (respectivamente), obtenidos en cada una de los entrenamientos, por cada red neuronal.

A continuación, se presenta los gráficos antes explicados para los resultados obtenidos con la CNN y posteriormente con la RNN.

CNN

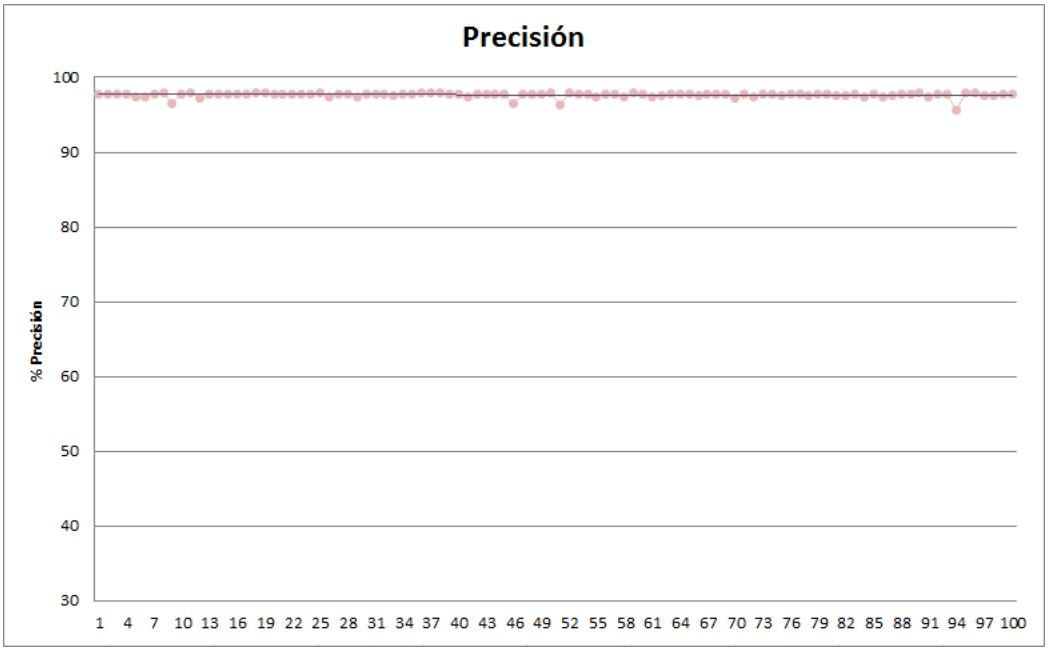

Figura 13. Exactitud por cada entrenamiento CNN

De los 100 entrenamientos de la arquitectura CNN, se obtuvo una precisión promedio del 97,68%. Se obtuvo además una desviación estándar de los valores de precisión de 0,33% respecto al promedio obtenido, lo que indica que esta red neuronal presenta una baja variabilidad en su precisión.

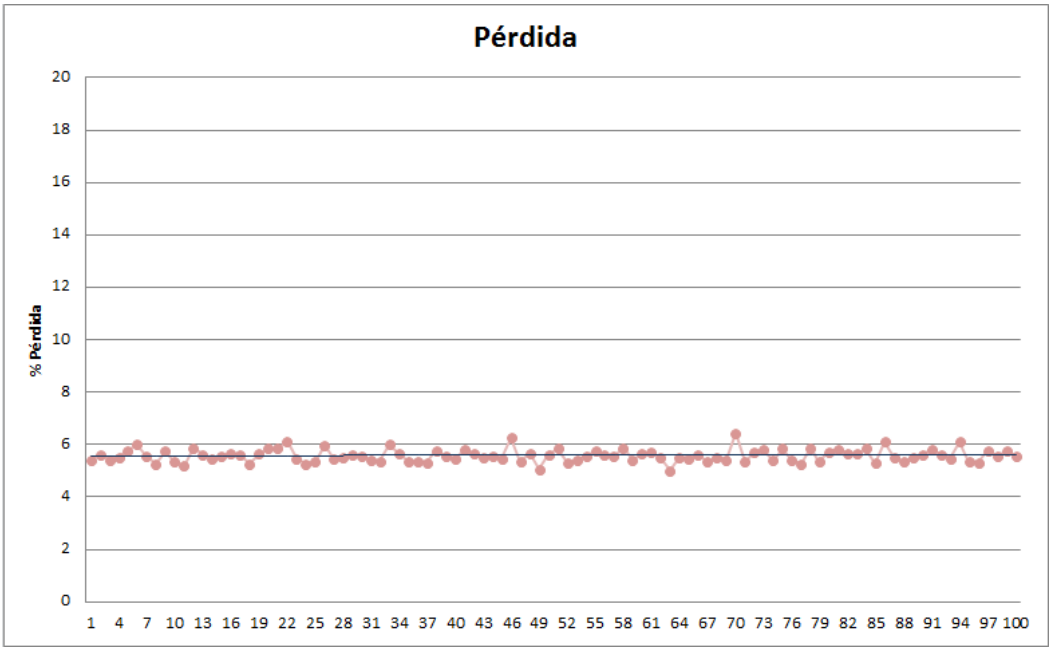

Figura 14. Pérdida por cada entrenamiento CNN

En cuanto a la pérdida de esta red neuronal, se obtuvo una pérdida promedio del 5,57%. La dispersión de las pérdidas respecto a la media es de 0,25%.

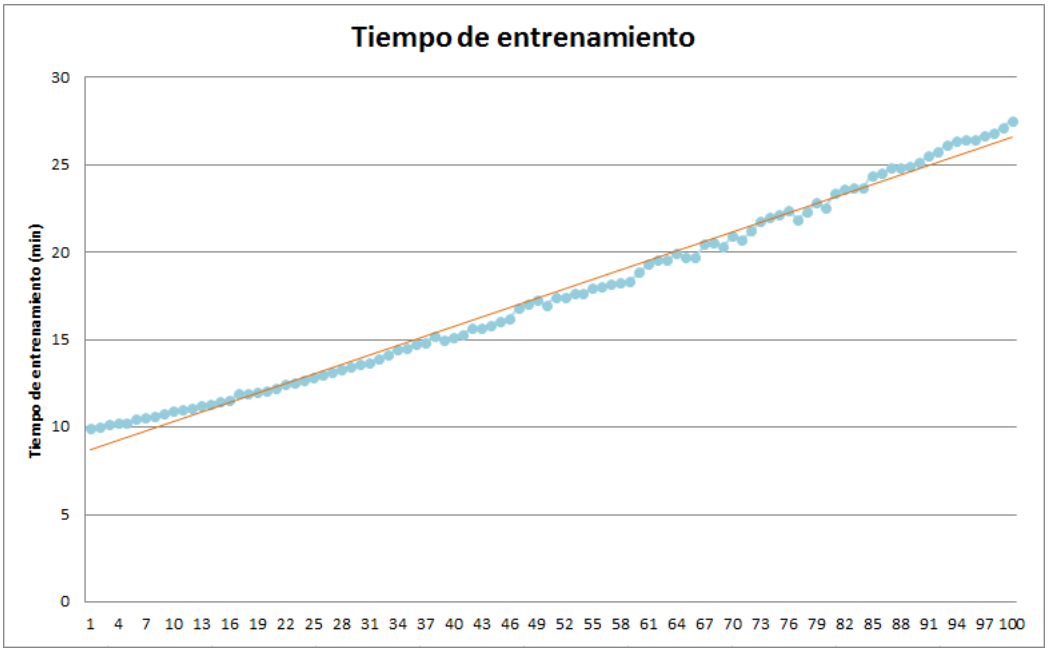

Figura 15. Tiempo por cada entrenamiento CNN

En cuanto al tiempo de duración obtenido en cada uno de los 100 entrenamientos, se observa una tendencia creciente comenzando con un tiempo de 9,95 minutos y terminando con 27,53 minutos. Se obtuvo un promedio de entrenamiento de 17,65 minutos y una desviación estándar de 5,25 minutos.

RNN

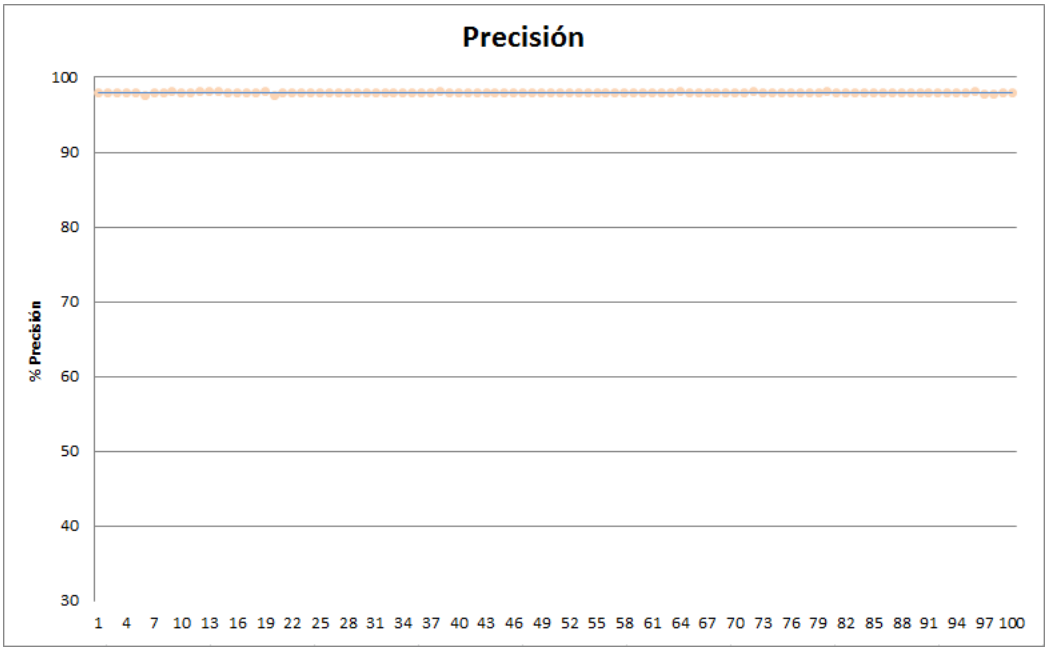

Figura 16. Exactitud por cada entrenamiento RNN

De los 100 entrenamientos de la arquitectura RNN, se obtuvo una precisión promedio del 97,96%. La desviación estándar de los valores de precisión es de 0,06% respecto al promedio obtenido, lo que indica que esta red neuronal también presenta una baja variabilidad en su precisión.

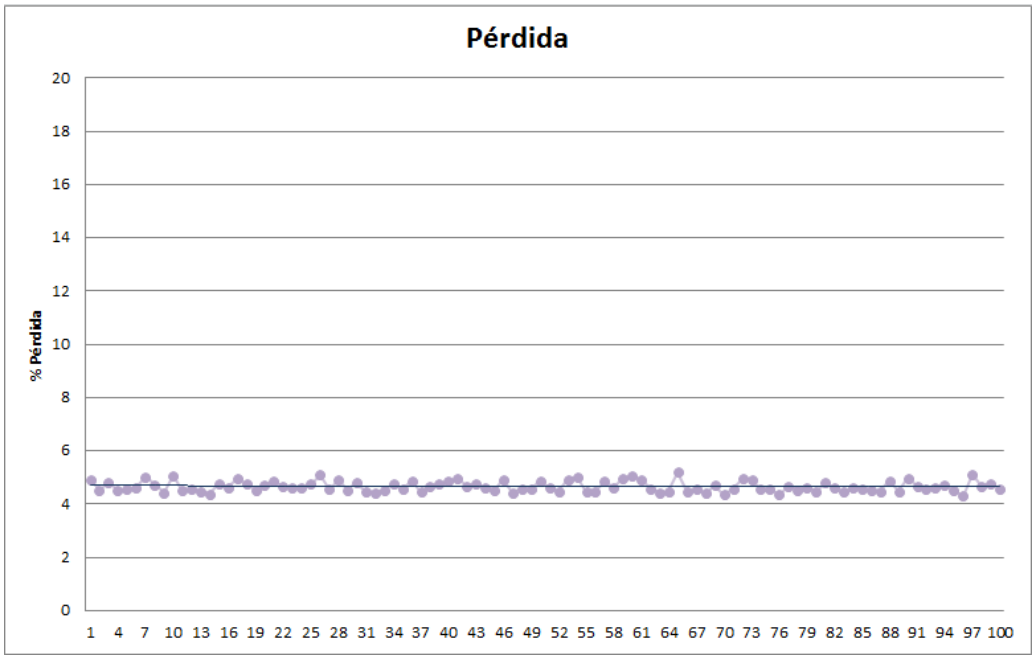

Figura 17. Pérdida por cada entrenamiento RNN

En cuanto a la pérdida de esta red neuronal, se obtuvo una pérdida promedio del 4,66%. La dispersión de las pérdidas respecto a la media es de 0,20%.

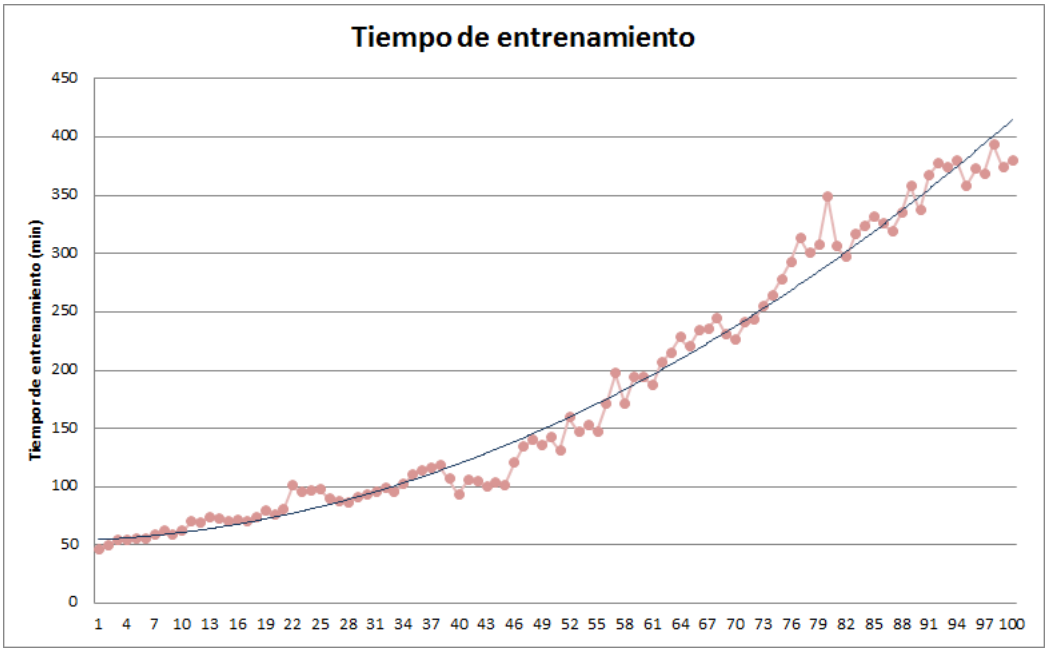

Figura 18. Tiempo por cada entrenamiento RNN

El tiempo de duración obtenido en cada uno de los 100 entrenamientos, se observa una tendencia creciente comenzando con un tiempo de 45,78 minutos y terminando en 379,72 minutos. Se obtuvo un promedio de 181,57 minutos y una desviación estándar de 109,34 minutos.

A continuación se presenta una tabla que resume los valores de precisión, pérdida y el tiempo de entrenamiento promedio de ambas redes neuronales. En la misma se resalta los mejores resultados obtenidos en cada valor de precisión, pérdida y el tiempo de entrenamiento promedio, además se indica los valores de la desviación estándar de cada resultado el cual no indica el grado de variabilidad de los mismos:

	Precisión (%)	**Perdida (%)**	**Tiempo de entrenamiento (min)**
CNN	97,68 +/- 0.33	5,58 +/- 0,25	17,65 +/- 5,25
RNN	97,96 +/- 0,06	4,66 +/- 0,2	181,57 +/- 109,34

Tabla 16. Resumen resultados de precisión, pérdida y tiempo de entrenamiento

4.3.Discusión

Los resultados obtenidos, permiten evidenciar que la red neuronal recurrente (RNN) obtuvo los mejores resultados en cuanto a la precisión y perdida promedio obtenida.

Es importante destacar que la precisión y pérdida de la red neuronal convolucional (CNN) no se aleja de los resultados obtenidos por la RNN, apenas existe una diferencia de 0,28% en la precisión y 0,91% en la perdida.

La pequeña ventaja de la RNN frente a la CNN, se debe principalmente a que la RNN es una red neuronal que presenta enlaces de retroalimentación, que le permiten que la información persista o se mantenga en la red durante algunas épocas del entrenamiento. Esta característica hace que esta red sea potente para el análisis de secuencias, como por ejemplo listas y datos temporales. Lo que la hace muy útil para el tipo de datos utilizado en esta investigación. A pesar de esta ventaja, la CNN en base a los resultados obtenidos

también demostró gran efectividad en la detección de características del conjunto de datos a través de sus capas convolucionales.

Los resultados obtenidos de precisión y pérdida en ambas redes neuronales presentan valores de desviación estándar bajos, es decir, que existe una baja variabilidad en los mismos. Esta baja variabilidad en los modelos implementados con estas redes neuronales permite inferir que los mismos son estables, ya que la estabilidad se define como algo que se mantiene sin peligro de cambiar a lo largo del tiempo.

En cuanto al tiempo de entrenamiento obtenido se valida que existe una amplia diferencia entre ambas redes neuronales. Es así que el tiempo de entrenamiento promedio de la CNN es aproximadamente diez veces menor que la RNN. La diferencia del tiempo de entrenamiento entre ambas redes neuronales, se debe principalmente a la cantidad de parámetros que fueron entrenados en cada red neuronal, como se observa en las siguientes tablas:

Capa (Tipo)	Forma de salida	Número de parámetros
Conv1D 1	(None, 2, 78)	6.084
Dropout 1	(None, 2, 78)	0
Max Pooling 1D 1	(None, 1, 78)	0
Flatten 1	(None, 78)	0
Dense 1	(None, 78)	6.162
Dropout 2	(None, 78)	0
Dense 2	(None, 13)	1.027
Dropout 3	(None, 13)	0
Dense 3	(None, 1)	14
Total de parámetros entrenables		13.287

Tabla 17. Tabla resumen del modelo CNN

Capa (Tipo)	Forma de salida	Número de parámetros
LSTM 1	(None, 1, 78)	48.984
LSTM 2	(None, 1, 156)	146.640
LSTM 3	(None, 13)	8.840
Dropout 1	(None, 13)	0
Dense 1	(None, 1)	14
Total de parámetros entrenables		204.478

Tabla 18. Tabla resumen del modelo RNN

Esta diferencia hace que la RNN requiere de un tiempo considerablemente mayor para entrenar todos los parámetros de la red.

Finalmente se puede observar en las figuras 15 y 18, como los tiempos de entrenamiento de ambas redes neuronales, tienen una tendencia creciente, es decir que no todos los entrenamientos tardan lo mismo, sino que cada entrenamiento tarda más que el anterior. Este comportamiento se debe a que los 100 entrenamientos de cada red neuronal fueron realizados en solo bucle, es decir, en un solo proceso. Lo que ocasionó que con el transcurso de los entrenamientos los recursos de memoria del computador que estaba ejecutando el proceso fuera progresivamente aumentando. A continuación, en las figuras 19 y 20, se puede observar un perfilamiento de la memoria en los entrenamientos tanto de la CNN como de la RNN, en donde claramente se observa lo comentado.

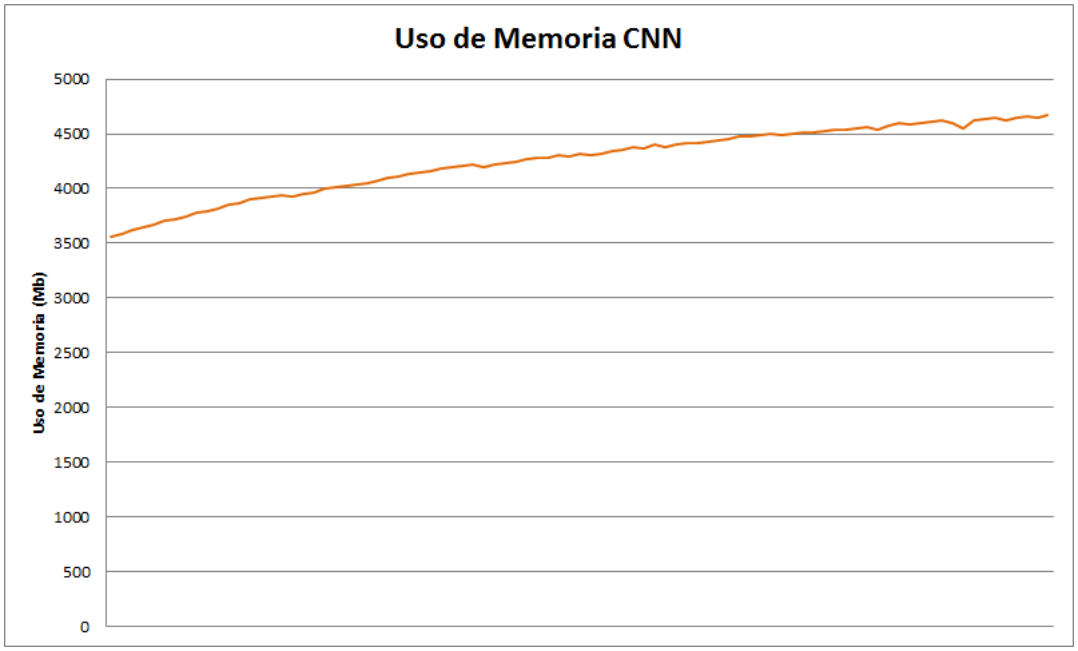

Figura 19. Uso de memoria entrenamientos CNN

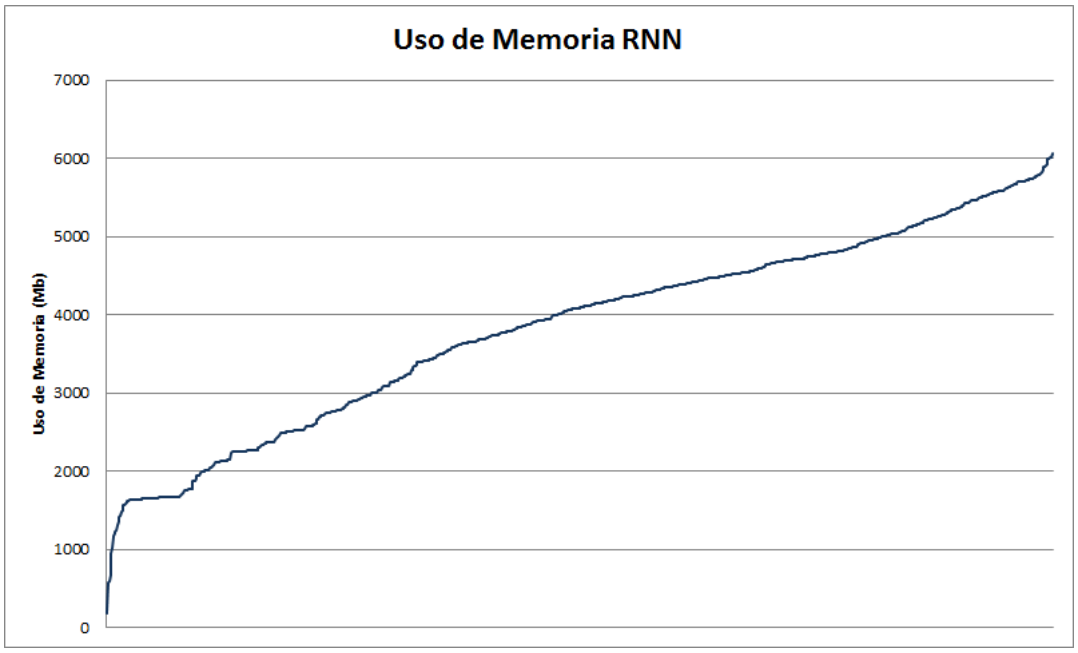

Figura 20. Uso de memoria entrenamientos RNN

En el perfilamiento de memoria de la CNN se aprecia un aumento de memoria con una variación de aproximadamente 1000 Mb en el transcurso de todo el proceso de

entrenamiento. Mientras que, en el perfilamiento de la RNN, se observa una variación mucho más pronunciada de aproximadamente 5000 Mb, es decir, que la RNN utilizó 5 veces más memoria que la CNN.

En ambos casos se puede indicar que la variación en el uso de la memoria puede explicar el aumento de los tiempos de entrenamiento de cada red neuronal. La ligera variación en el uso de memoria de la CNN puede dar razón a la linealidad observada en los tiempos de entrenamiento de esta red neuronal, mientras que la pronunciada variación de la RNN puede ser causa de la tendencia exponencial de los tiempos de entrenamiento. Con este análisis se puede indicar que existe una relación casi directa entre el tiempo de entrenamiento y los recursos de memoria del computador.

CAPÍTULO V

CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

Este trabajo de investigación, presenta un aporte para el desarrollo de sistemas de detección de tráfico malicioso en una red de datos, mediante el aprovechamiento de los modelos de predicción desarrollados usando redes neuronales convolucionales y recurrentes, generados en función del conjunto de datos CICIDS2017.

El conjunto de datos CICIDS2017, fue esencial para el proceso de entrenamiento de ambas redes neuronales. Se destaca la importancia del pre procesamiento del mismo debido a que se presentó inconsistencias en los tipos de datos, además de la existencia de datos especiales., El conjunto de datos pre procesado constituyó un pilar esencial para la preparación de los modelos.

El estudio documental realizado sobre las redes neuronales convolucionales y recurrentes, determinando sus principales características, fue determinante para la elección de las arquitecturas de cada modelo y su posterior codificación en Python. Este lenguaje de programación y su librería *keras*, permitieron que la programación de los modelos sea rápida e intuitiva, gracias a la construcción de las redes neuronales como una secuencia de capas.

El proceso de entrenamiento y aprendizaje de ambas redes neuronales, se realizó utilizando técnicas de aprendizaje profundo. La CNN, mediante el uso de las capas convolucionales, permitió la extracción y comprensión de las características de cada flujo de datos, lo que garantizó que el procesamiento del modelo sea eficiente en tiempo de entrenamiento, y que las predicciones obtenidas sean precisas. Mientras que la RNN, gracias a sus bucles de retro alimentación en cada una de sus capas, demostró su eficiencia

en el manejo de datos secuenciales. El procesamiento de este modelo resultó ser preciso en sus predicciones, pero costoso en cuanto al tiempo de entrenamiento y el uso de recursos de memoria.

El uso de la técnica de validación cruzada aleatoria, empleado en la función *evaluate*, permitió verificar que los modelos para ambas redes neuronales obtuvieron un promedio de precisión sobre el 97%y una pérdida menor al 6%. En cuanto al tiempo de entrenamiento de ambas redes neuronales, se observó que la RNN tardó aproximadamente 10 veces más que la CNN

Los datos antes mencionados permitieron concluir que si consideramos solamente la precisión obtenida como referencia para determinar cuál red neuronal obtuvo el mejor desempeño en la detección de tráfico malicioso, la red neuronal recurrente sería la seleccionada. Ahora bien, si se considera la amplia diferencia observada en el tiempo de entrenamiento y la mínima diferencia que existe entre las medidas de precisión entre ambas redes neuronales, no se puede descartar que la CNN pueda ser considera como la red de mejor desempeño en la detección de tráfico malicioso, y la selección entre ambas dependerá de si la aplicación en la cual se pretenda utilizar el modelo el tiempo de entrenamiento de la red neuronal sea un factor determinante.

5.2.Trabajos Futuros

El estudio realizado en este trabajo de investigación, da pautas para que nazcan nuevas iniciativas en la contribución para el uso de técnicas o el desarrollo de aplicaciones que estén enfocadas en la detección de intrusos en una red de datos. De esta forma varios trabajos pueden surgir como consecuencia del desarrollo de esta tesis:

- Utilizando el conjunto de datos CICIDS2017 es posible también generar nuevos modelos de predicción que detecten un tipo de ataque específico, considerando que el conjunto de datos CICIDS2017 tiene identificado y etiquetado tráfico de red para doce tipos de ataques diferentes.
- Realizar predicciones con los modelos generados, utilizando para este fin un conjunto de datos capturado en un entorno de red en producción, como puede ser el caso de una red empresarial.

- Incluir los modelos generados en este estudio en una aplicación en tiempo real que capture el tráfico de una red de datos y permita detectar tráfico malicioso en la misma. Utilizando para esto técnicas de búsqueda de arquitecturas neuronales (NAS), con el fin de hacer un re-diseño de los modelos propuestos en este estudio, y validar si a través de esta técnica es posible mejorar la precisión, eficiencia y tiempo de entrenamiento de los modelos.
- Realizar un análisis comparativo de las redes neuronales utilizadas en este estudio, con otros modelos de aprendizaje profundo como *autoencoders*, y con modelos de aprendizaje de máquina supervisados como el caso de perceptrones multicapa, arboles de decisión, *k*-nearest *neighbours*, redes neuronales artificiales, redes bayesianas, *random forest*, *support vector machine*, etc. Utilizando para este análisis el conjunto de datos CICIDS2017.

REFERENCIAS BIBLIOGRÁFICAS

Abadeh, M. S., Mohamadi, H., & Habibi, J. (2011). Design and analysis of genetic fuzzy systems for intrusion detection in computer networks. *Expert Systems with Applications*, *38*(6), 7067–7075.

Abdullah, M., Alshannaq, A., Balamash, A., & Almabdy, S. (2018). Enhanced Intrusion Detection System using Feature Selection Method and Ensemble Learning Algorithms. *International Journal of Computer Science and Information Security (IJCSIS)*, *16*(2).

Abdulrahman, A. A., & Ibrahem, M. K. (2019). Evaluation of DDoS attacks Detection in a New Intrusion Dataset Based on Classification Algorithms. *Iraqi Journal of Information & Communications Technology*, *1*(3), 49–55.

Aburomman, A. A., & Ibne Reaz, M. Bin. (2016). A novel SVM-kNN-PSO ensemble method for intrusion detection system. *Applied Soft Computing*, *38*, 360–372.

Abuzneid, A. A., Faezipour, M., Abdulhammed, R., Abu Mallouh, A., & Musafer, H. (2019). Machine Learning Based Feature Reduction for Network Intrusion Detection.

Ackermann, N. (2018a). Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences. Retrieved June 11, 2019, from https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf

Ackermann, N. (2018b). Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences. Retrieved March 25, 2019, from https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf

Aguilera, P. (2011). *Redes seguras (Seguridad informática)*. (Editex, Ed.) (1st ed.).

Ahmad, D., & Hanapi, Z. (2013). Hybrid of Fuzzy Clustering Neural Network Over NSL Dataset for Intrusion Detection System. *Journal of Computer Science*, *9*(3), 391–403.

Ahmim, A., Ferrag, M., Maglaras, L., Derdour, M., & Janicke, H. (2019). A Detailed Analysis of Using Supervised Machine Learning for Intrusion Detection.

Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., & Janicke, H. (2019). A Novel Hierarchical Intrusion Detection System based on Decision Tree and Rules-based Models. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)* (pp. 228–233). IEEE.

Akhi, A., Kanon, E., Kabir, A., & Banu, A. (2019). *Network Intrusion Classification Employing Machine Learning : A Survey (Doctoral dissertation)*.

Aksu, Dogukan, & Ali Aydin, M. (2018). Detecting Port Scan Attempts with Comparative Analysis of Deep Learning and Support Vector Machine Algorithms. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)* (pp. 77–80). IEEE.

Aksu, Doğukan, Üstebay, S., Aydin, M. A., & Atmaca, T. (2018). Intrusion Detection with Comparative Analysis of Supervised Learning Techniques and Fisher Score Feature Selection Algorithm. In *International Symposium on Computer and Information Sciences* (pp. 141–149). Springer, Cham.

AL-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*, *41*, 1–11.

Ali, M. H., Al Mohammed, B. A. D., Ismail, A., & Zolkipli, M. F. (2018). A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization. *IEEE Access*, *6*, 20255–20261.

Alipour, H., Al-Nashif, Y. B., Satam, P., & Hariri, S. (2015). Wireless Anomaly Detection Based on IEEE 802.11 Behavior Analysis. *IEEE Transactions on Information Forensics and Security*, *10*(10), 2158–2170.

Ambusaidi, M. A., He, X., Nanda, P., & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Transactions on Computers*, *65*(10), 2986–2998.

Amoh, J., & Odame, K. (2016). Deep Neural Networks for Identifying Cough Sounds. *IEEE Transactions on Biomedical Circuits and Systems*, *10*(5), 1003–1011.

Anderson, J. P. (1980). *Computer security threat monitoring and surveillance*. *Technical Report James P. Anderson Co. Fort Washington Pa.* James P. Anderson Co.

Aneetha, A., & Bose, B. (2012). The Combined Approach for Anomaly Detection Using Neural Networks and Clustering Techniques. *Computer Science & Engineering: An International Journal*, *2*(4), 37–46.

Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., & Ebrahimi, A. (2016). A hybrid method consisting of GA and SVM for intrusion detection system. *Neural Computing and Applications*, *27*(6), 1669–1676.

Azwar, H., Murtaz, M., Siddique, M., & Rehman, S. (2018). Intrusion Detection in secure network for Cybersecurity systems using Machine Learning and Data Mining. In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)* (pp. 1–9). IEEE.

Bae, S. H., Choi, I., & Kim, N. S. (2016). Acoustic Scene Classification Using Parallel Combination of LSTM and CNN. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, (September), 11–15.

Bahrololum, M., Salahi, E., & Khaleghi, M. (2009). Anomaly Intrusion Detection Design Using Hybrid of Unsupervised and Supervised Neural Network. *International Journal of Computer Networks & Communications (IJCNC)*, *1*(2), 26–33.

Baig, Z. A., Sait, S. M., & Shaheen, A. (2013). GMDH-based networks for intelligent intrusion detection. *Engineering Applications of Artificial Intelligence*, *26*(7), 1731–1740.

Balamurugan, V., & Saravanan, R. (2017). Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and OTS generation. *Cluster Computing*, 1–13.

Bansal, A., & Kaur, S. (2018). Extreme Gradient Boosting Based Tuning for Classification in Intrusion Detection Systems. In *International Conference on Advances in Computing and Data Sciences* (pp. 372–380). Springer, Singapore.

Bayer, U., Moser, A., Kruegel, C., & Kirda, E. (2006). Dynamic analysis of malicious code. *Journal in Computer Virology*, *2*(1), 67–77.

Bellantonio, M., Haque, M. A., Rodriguez, P., Nasrollahi, K., Telve, T., Escalera, S., … Anbarjafari, G. (2017). Spatio-temporal Pain Recognition in CNN-Based Super-Resolved Facial Images. In *Video Analytics. Face and Facial Expression Recognition and Audience Measurement: Third International Workshop, VAAM 2016, and Second International Workshop, FFER 2016, Cancun, Mexico, December 4, 2016, Revised Selected Papers* (Vol. 10165, p. 151). Springer.

Bertero, D., & Fung, P. (2016). Deep Learning of Audio and Language Features for Humor Prediction. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* (pp. 496–501).

Bertona, L. (2005). *Entrenamiento de redes neuronales basado en algoritmos evolutivos*. Doctoral dissertation, UNIVERSIDAD DE BUENOS AIRES.

Bouzida, Y., & Cuppens, F. (2004). Efficient intrusion detection using principal component analysis. In *3éme Conférence sur la Sécurité et Architectures Réseaux (SAR), La Londe, France* (pp. 381–395).

Bulavas, V. (2018). Applying Machine Learning Methods for Early Detection of Cyber

Security Incidents, 1–15.

Cakir, E., Parascandolo, G., Heittola, T., Huttunen, H., & Virtanen, T. (2017). Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(6), 1291–1303.

Capó, L., & García, D. (2015). *Sistemas de Detección de Intrusos en Seguridad Informática*.

Carreras, A. (2018). *QBarrier : Control de acceso a parking utilizando aprendizaje profundo*. Tesis de Licenciatura.

Catania, C., Bromberg, F., & García, C. (2012). An autonomous labeling approach to SVM algorithms for network traffic anomaly detection. *Proceedings of ASAI 2009 Argentine Symposium on Artificial Intelligence*, (January), 144–158.

Chandrashekhar, A. ., & Raghuveer. (2013). Fortification of Hybrid Intrusion Detection System Using Variants of Neural Networks and Support Vector Machines. *International Journal of Network Security & Its Applications*, *5*(1), 71–90.

Chavan, S., Shah, K., Dave, N., Mukherjee, S., Abraham, A., & Sanyal, S. (2004). Adaptive neuro-fuzzy intrusion detection systems. In *International Conference on Information Technology: Coding Computing, ITCC* (Vol. 1, pp. 70–74).

Chawla, A., Lee, B., Fallon, S., & Jacob, P. (2019). Host Based Intrusion Detection System with Combined CNN/RNN Model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 149–158). Springer, Cham.

Chen, W.-H., Hsu, S.-H., & Shen, H.-P. (2005). Application of SVM and ANN for intrusion detection. *Computers & Operations Research*, *32*(10), 2617–2634.

Chen, Y., Abraham, A., & Yang, B. (2007). Hybrid flexible neural-tree-based intrusion detection systems. *International Journal of Intelligent Systems*, *22*(4), 337–352.

Cheriyan, E., Nandakumar, M. P., & Sindhu, K. C. (2010). RTRL Algorithm Based Adaptive Controller for Non-linear Multivariable Systems. *International Journal of Computer Applications*, *975*, 8887.

Chi Cheng, Wee Peng Tay, & Huang, G.-B. (2012). Extreme learning machines for intrusion detection. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE.

Chimphlee, W., Abdullah, A., Sap, M. N. M., Srinoy, S., & Chimphlee, S. (2006). Anomaly-based intrusion detection using fuzzy rough clustering. In *Proceedings - 2006 International Conference on Hybrid Information Technology, ICHIT 2006* (Vol. 1, pp. 329–334).

Cruz, B., Martínez, S., Abed, R., Ábalo, G., Lorenzo, G., Matilde, M., … Lorenzo, M. G. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*, *1*(4), 48–57.

Cui, J., Long, J., Min, E., Liu, Q., & Li, Q. (2018). Comparative Study of CNN and RNN for Deep Learning Based Intrusion Detection System. In *International Conference on Cloud Computing and Security* (pp. 159–170). Springer, Cham.

Das, K., Schneider, J., & Neill, D. B. (2008). Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 169–176). ACM.

Davanzo, G., Medvet, E., & Bartoli, A. (2011). Anomaly detection techniques for a web defacement monitoring service. *Expert Systems with Applications*, *38*(10), 12521–12530.

Dave, K. (2013). Brute-force Attack “Seeking but Distressing.” *International Journal of Innovations in Engineering and Technology*, *2*(3), 75–78.

Deng, L., & Platt, J. (2014). Ensemble Deep Learning for Speech Recognition. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Denning, D. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, (2), 222–232.

Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, *29*(4), 713–722.

Devaraju, S., & Ramakrishnan, S. (2013). Detection of Accuracy for Intrusion Detection System Using Neural Network, *3*(1), 338–345.

Díaz, J., & Salcedo, J. (2014). Sistema de Prevención de Intrusos para mejorar la seguridad de los servidores de la Universidad Nacional de Trujillo, 52.

Du, T., & Shanker, V. K. (2013). *Deep Learning for Natural Language Processing*. Springer, Singapore.

Duan, L., & Xiao, Y. (2018). An Intrusion Detection Model Based on Fuzzy C-means Algorithm. *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 120–123.

Durán, J. (2017). Redes neuronales convolucionales en R: Reconocimiento de caracteres escritos a mano.

Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., … Paxson, V. (2014). The Matter of Heartbleed. In *Proceedings of the 2014 conference on internet measurement conference* (pp. 475–488). ACM.

Eesa, A. S., Orman, Z., & Brifcani, A. M. A. (2015). A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Systems with Applications*, *42*(5), 2670–2679.

Effendy, D. A., Kusrini, K., & Sudarmawan, S. (2017). Classification of intrusion detection system (IDS) based on computer network. In *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*

(pp. 90–94). IEEE.

Eskin, E., Arnold, A., Prerau, M., & Portnoy, L. (2002). A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, *3*(40).

Evans, D. (2011). *The Internet of Things: How the Next Evolution of the Internet is Changing Everything*. *Tech. Rep.*

Fadlullah, Z. M., Nishiyama, H., Kato, N., & Fouda, M. M. (2013). Intrusion detection system (IDS) for combating attacks against cognitive radio networks. *IEEE Network*, *27*(3), 51–56.

Fan, W., Miller, M., Stolfo, S., Lee, W., & Chan, P. (2004). Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, *6*(5), 507–527.

Farid, D., Rahman, M., & Rahman, C. (2011). Adaptive Intrusion Detection based on Boosting and Naive Bayesian Classifier. *International Journal of Computer Applications*, *24*(3), 12–19.

Feng, W., Zhang, Q., Hu, G., & Huang, J. X. (2014). Mining network data for intrusion detection through combining SVMs with ant colony networks. *Future Generation Computer Systems*, *37*, 127–140.

Forrest, S., Hofmeyr, S., Somayaji, A., & Longstaff, T. (1996). A Sense of Self for Unix Processes, 120–128.

Gaikwad, D., Jagtap, S., Thakare, K., & Budhawant, V. (2012). Anomaly Based Intrusion Detection System Using Artificial Neural Network and Fuzzy Clustering. *International Journal of Engineering Research & Technology (IJERT)*, *1*(9).

García, B. (2015). Implementación de Técnicas de Deep Learning, 49.

Garcia, J., & Navarro, G. (2009). A Survey on Cross-Site Scripting Attacks.

Gers, F. (2001). *Long Short-Term Memory in Recurrent Neural Networks*. Doctoral dissertation, Verlag nicht ermittelbar.

Gestal, M. (2013). Introducción a los Algoritmos Genéticos. *Univesidad de Coruña*.

Giacinto, G., & Roli, F. (2003). Intrusion detection in computer networks by multiple classifier systems. In *Object recognition supported by user interaction for service robots* (Vol. 2, pp. 390–393). IEEE.

Giacinto, Giorgio, Perdisci, R., Del Rio, M., & Roli, F. (2008). Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, *9*(1), 69–82.

Gibert, D. (2016). Convolutional Neural Networks for Malware Classification. *University Rovira i Virgili, Tarragona, Spain*.

González, A. (2018). *Aplicaciones de técnicas de intelignecia artificial basadas en aprendizaje profundo (deep learning) al análisis y mejora de la eficiencia de procesos industriales*.

González, C. (2015). Lógica Difusa: Una introducción práctica.

González, D. (2003). Sistemas de Detección de Intrusos.

Gordo, R., Gutiérrez, S., & Rodríguez, P. (2013). *Sistema Concurrente de Detección de Intrusiones con Correlación de Alertas en Entornos Distribuidos*.

Grinblat, G. L., Uzal, L. C., & Granitto, P. M. (2013). Abrupt change detection with One-Class Time-Adaptive Support Vector Machines. *Expert Systems with Applications*, *40*(18), 7242–7249.

Grosse, R. (2017). Exploding and Vanishing Gradients. *University of Toronto Computer Science*, 1–11.

Gunes, H., Nur, A., & Heywood, M. I. (2007). A hierarchical SOM-based intrusion

detection system. *Engineering Applications of Artificial Intelligence*, *20*(4), 439–451.

Guo, Y., Liu, Y., Bakker, E. M., Guo, Y., & Lew, M. S. (2018). CNN-RNN: a large-scale hierarchical image classification framework. *Multimedia Tools and Applications*, *77*(8), 10251–10271.

Hadri, A., Chougdali, K., & Touahni, R. (2016). Intrusion detection system using PCA and Fuzzy PCA techniques. In *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)* (pp. 1–7). IEEE.

He, D., Chen, X., Zou, D., Pei, L., & Jiang, L. (2018). An Improved Kernel Clustering Algorithm Used in Computer Network Intrusion Detection. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1–5). IEEE.

Heller, K., Svore, K., Keromytis, A., & Stolfo, S. (2003). One class support vector machines for detecting anomalous windows registry accesses. *Workshop on Data Mining for Computer Security (DMSEC).*

Hernández, C., & Rodríguez, J. (2008). Preprocesamiento de datos estructurados. *Vínculos*, *4*(2), 27–48.

Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2018). A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets, *1*(1).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P.-L., Iorkyase, E., Tachtatzis, C., & Atkinson, R. (2016). Threat analysis of IoT networks using artificial neural network intrusion detection system. In *2016 International Symposium on Networks, Computers and Communications (ISNCC)* (pp. 1–6). IEEE.

Hodo, E., Bellekens, X., Iorkyase, E., Hamilton, A., Tachtatzis, C., & Atkinson, R. (2017).

Machine Learning Approach for Detection of nonTor Traffic. In *Proceedings of the 12th International Conference on Availability, Reliability and Security* (p. 85). ACM.

Hosseini Bamakan, S. M., Wang, H., Yingjie, T., & Shi, Y. (2016). An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization. *Neurocomputing*, *199*, 90–102.

Hou, Y.-T., Chang, Y., Chen, T., Laih, C.-S., & Chen, C.-M. (2010). Malicious web content detection by machine learning. *Expert Systems with Applications*, *37*(1), 55–60.

Hsu, C.-M., Hsieh, H.-Y., Prakosa, S. W., Azhari, M. Z., & Leu, J.-S. (2018). Using Long-Short-Term Memory Based Convolutional Neural Networks for Network Intrusion Detection. In *11th EAI International Conference, WiCON 2018* (pp. 86–94). Taipei, Taiwan.

Huang, C.-W., Chiang, C.-T., & Li, Q. (2017). A study of deep learning networks on mobile traffic forecasting. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (pp. 1–6). IEEE.

Idika, N., & Mathur, A. (2007). A survey of Malware Detection Techniques. *Purdue University*, *48*, 2–48.

Internet Society. (2016). Global Internet Report 2016. *Internet Society*, .

Jabez, J., & Muthukumar, B. (2015). Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach. *Procedia Computer Science*, *48*, 338–346.

Jawhar, M., & Mehrotra, M. (2010). Design Network Intrusion Detection System using hybrid Fuzzy-Neural Network. *International Journal of Computer Science and Security*, *4*(3), 285–294.

Jeon, S., Shin, J.-W., Lee, Y.-J., Kim, W.-H., Kwon, Y., & Yang, H.-Y. (2017). Empirical study of drone sound detection in real-life environment with deep neural networks. In

2017 25th European Signal Processing Conference (EUSIPCO) (pp. 1858–1862). IEEE.

Ji, S.-Y., Jeong, B.-K., Choi, S., & Jeong, D. H. (2016). A multi-level intrusion detection method for abnormal network behaviors. *Journal of Network and Computer Applications*, *62*, 9–17.

Joo, D., Hong, T., & Han, I. (2003). The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. *Expert Systems with Applications*, *25*(1), 69–75.

Kang, D., Fuller, D., & Honavar, V. (2005). Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation. *Workshop on Information Assurance and Security*, 511–516.

Kang, I., Jeong, M. K., & Kong, D. (2012). A differentiated one-class classification method with applications to intrusion detection. *Expert Systems with Applications*, *39*(4), 3899–3905.

Kaur, S., & Singh, M. (2019). Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks. *Neural Computing and Applications*, 1–19.

Khan, L., Awad, M., & Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. *VLDB Journal*, *16*(4), 507–521.

Khosla, E., Ramesh, D., Sharma, R. P., & Nyakotey, S. (2018). RNNs-RT: Flood based Prediction of Human and Animal deaths in Bihar using Recurrent Neural Networks and Regression Techniques. *Procedia Computer Science*, *132*(August), 486–497.

Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, *41*(4), 1690–1700.

Kindy, D. A., & Pathan, A.-S. K. (2011). A survey on SQL injection: Vulnerabilities,

attacks, and prevention techniques. In *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)* (pp. 468–471). IEEE.

Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Systems with Applications*, *39*(18), 13492–13500.

Kolter, J., & Maloof, M. (2004). Learning to detect malicious executables in the wild. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04* (Vol. 7, p. 470).

Kong, D., & Yan, G. (2013). Discriminant malware distance learning on structuralinformation for automated malware classification. *ACM SIGMETRICS Performance Evaluation Review*, *41*(1), 347.

Kruegel, C., Valeur, F., & Vigna, G. (2005). *Intrusion Detection and Correlation - Challenges and Solutions* (1st ed.). SpringerUS.

Kumar, A., Kaur, P., & Kumar, S. (2012). Madam Id for Intrusion Detection Using Data Mining. *International Journal of Research in IT & Management*, *2*(2), 256–263.

Kumar, R., & Sharma, D. (2018). HyINT: Signature-Anomaly Intrusion Detection System. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1–7). IEEE.

Kumar Shrivas, A., & Kumar Dewangan, A. (2014). An Ensemble Model for Classification of Attacks with Feature Selection based on KDD99 and NSL-KDD Data Set. *International Journal of Computer Applications*, *99*(15), 8–13.

Lago, J., De Ridder, F., & De Schutter, B. (2018). Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, *221*, 386–405.

Lee, B., Amaresh, S., Green, C., & Engels, D. (2018). Comparative Study of Deep

Learning Models for Network Intrusion Detection. *SMU Data Science Review*, *1*(1), 8.

Lee, S., Kim, G., & Kim, S. (2011). Self-adaptive and dynamic clustering for online anomaly detection. *Expert Systems with Applications*, *38*(12), 14891–14898.

Lee, T., & Mody, J. J. (2006). *Behavioral Classification*. *Proceedings of the European Institute for Computer Antivirus Research Conference (EICAR'06)*.

Lee, W., Stolfo, S., & Mok, K. (1999). A data mining framework for building intrusion. *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium On*, 120–132.

Li, Q., Tan, Z., Jamdagni, A., Nanda, P., He, X., & Han, W. (2017). An Intrusion Detection System Based on Polynomial Feature Correlation Analysis. In *2017 IEEE Trustcom/BigDataSE/ICESS* (pp. 978–983). IEEE.

Li, W., Meng, W., Kwok, L.-F., & IP, H. H. S. (2017). Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model. *Journal of Network and Computer Applications*, *77*, 135–145.

Li, Yang, & Guo, L. (2007). An active learning based TCM-KNN algorithm for supervised network intrusion detection. *Computers & Security*, *26*(7–8), 459–467.

Li, Yinhui, Xia, J., Zhang, S., Yan, J., Ai, X., & Dai, K. (2012). An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications*, *39*(1), 424–430.

Liao, Y., & Vemuri, V. R. (2002). Use of k-nearest neighbor classifier for intrusion detection. *Computers and Security*, *21*(5), 439–448.

Lin, C. T., & Lee, C. S. G. (1996). Neural fuzzy systems : a neuro-fuzzy synergism to intelligent systems, *205*.

Lin, S.-W., Ying, K.-C., Lee, C.-Y., & Lee, Z.-J. (2012). An intelligent algorithm with

feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, *12*(10), 3285–3290.

Lin, W.-C., Ke, S.-W., & Tsai, C.-F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, *78*, 13–21.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., … Zissman, M. A. (2000). Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00* (Vol. 2, pp. 12–26). IEEE.

Lisehroodi, M., Muda, Z., & Yassin, W. (2013). A Hybrid Framework Based on Neural Network Mlp and K-Means Clustering for Intrusion Detection System. *Computing & Informatics, 4Th International Conference, 2013*, (020), 305–311.

Liu, B., Shi, L., Cai, Z., & Li, M. (2012). Software Vulnerability Discovery Techniques: A Survey. In *2012 Fourth International Conference on Multimedia Information Networking and Security* (pp. 152–156). IEEE.

Liu, G., & Yi, Z. (2006). Intrusion Detection Using PCASOM Neural Networks. In *International Symposium on Neural Networks* (pp. 240–245). Springer, Berlin, Heidelberg.

Liu, G., Yi, Z., & Yang, S. (2007). A hierarchical intrusion detection model based on the PCA neural networks. *Neurocomputing*, *70*(7–9), 1561–1568.

Liu, S., Hu, X., & Wang, W. (2018). Deep Reinforcement Learning Based Dynamic Channel Allocation Algorithm in Multibeam Satellite Systems. *IEEE Access*, *6*, 15733–15742.

Liu, Y., Chen, K., Liao, X., & Zhang, W. (2004). A genetic clustering method for intrusion detection. *Pattern Recognition*, *37*(5), 927–942.

Lo Bosco, G., & Di Gangi, M. A. (2017). Deep Learning Architectures for DNA Sequence Classification. In *Fuzzy Logic and Soft Computing Applications* (pp. 162–171). Springer, Cham.

Lopez, A., Mohan, A., & Nair, S. (2019). Network Traffic Behavioral Analytics for Detection of DDoS Attacks. *SMU Data Science Review*, *2*(1), 14.

Lorenzo, I., Maciá, F., Mora, J., Marcos, D., Gil, J., & Lau, R. (2009). Sistema de Detección de Intrusos de Red basado en Técnicas de Aprendizaje de Máquina: Revisión Bibliográfica.

Lu, W., & Tong, H. (2009). Detecting Network Anomalies Using CUSUM and EM Clustering. In *International Symposium on Intelligence Computation and Applications* (pp. 297–308). Springer, Berlin, Heidelberg.

Luna, J. (2015). *SISTEMA DETECTOR DE INTRUSIONES OCUPANDO UNA RED NEURONAL ARTIFICIAL*. (Tesis de Maestría. Universidad Autónoma del Estado de México).

Luo, J., & Bridges, S. M. (2000). Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems*, *15*(8), 687–703.

Masduki, B. W., Ramli, K., Saputra, F. A., & Sugiarto, D. (2015). Study on implementation of machine learning methods combination for improving attacks detection accuracy on Intrusion Detection System (IDS). In *2015 International Conference on Quality in Research (QiR)* (pp. 56–64). IEEE.

Matich, D. (2001). Redes Neuronales: Conceptos Básicos y Aplicaciones. *Universidad Tecnológica Nacional, México*, 6.

McKay, R., Pendleton, B., Britt, J., & Nakhavanit, B. (2019). Machine Learning Algorithms on Botnet Traffic: Ensemble and Simple Algorithms. In *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis - ICCDA 2019*

(pp. 31–35). New York, USA: ACM Press.

Min, E., Long, J., Liu, Q., Cui, J., Cai, Z., & Ma, J. (2018). SU-IDS: A Semi-supervised and Unsupervised Framework for Network Intrusion Detection. In *International Conference on Cloud Computing and Security* (pp. 322–334). Springer, Cham.

Mitchel, T. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math; (March 1, 1997).

MkDocs. (2014). Keras Documentation. Retrieved April 30, 2019, from https://keras.io/

Moar, J. (2018). *The Future of Cybercrime & Security: Threat Analysis, Impact Assessment & Leading Vendors 2019-2024*.

Mohammed, M., & Pathan, A.-S. (2016). Intrusion Detection and Prevention Systems (IDPSs). In *Automatic Defense Against Zero-day Polymorphic Worms in Communication Networks* (pp. 47–84).

Mok, M. S., Sohn, S. Y., & Ju, Y. H. (2010). Random effects logistic regression model for anomaly detection. *Expert Systems with Applications*, *37*(10), 7162–7166.

Moradi, M., & Zulkernine, M. (2004). A Neural Network Based System for Intrusion Detection and Classification of Attacks. In *Proceedings of the IEEE International Conference on Advances in Intelligent Systems-Theory and Applications* (pp. 15–18).

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal*, *25*(1–3), 18–31.

Muda, Z., Yassin, W., Sulaiman, M. N., & Udzir, N. I. (2011). Intrusion detection based on K-Means clustering and Naïve Bayes classification. In *2011 7th International Conference on Information Technology in Asia* (pp. 1–6). IEEE.

Mukkamala, S., Sung, A., & Abraham, A. (2005). Intrusion detection using an ensemble of

intelligent paradigms. *Journal of Network and Computer Applications*, *28*(2), 167–182.

Nagamani, C., & Chittineni, S. (2018). Network Intrusion Detection Mechanisms Using Outlier Detection. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 1468–1473). IEEE.

Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011). Malware Images: Visualization and Automatic Classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec '11* (pp. 1–7).

Ng, J., Joshi, D., & Banik, S. M. (2015). Applying Data Mining Techniques to Intrusion Detection. In *2015 12th International Conference on Information Technology - New Generations* (pp. 800–801). IEEE.

Nskh, P., Varma, M. N., & Naik, R. R. (2016). Principle component analysis based intrusion detection system using support vector machine. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)* (pp. 1344–1350). IEEE.

Özyer, T., Alhajj, R., & Barker, K. (2007). Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening. *Journal of Network and Computer Applications*, *30*(1), 99–113.

Pan, S., Morris, T., & Adhikari, U. (2015). Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems. *IEEE Transactions on Smart Grid*, *6*(6), 3104–3113.

Park, S., & Park, H. (2019). ANN Based Intrusion Detection Model. In *Workshops of the International Conference on Advanced Information Networking and Applications* (pp. 433–437). Springer, Cham.

Peddabachigari, S., Abraham, A., Grosan, C., & Thomas, J. (2007). Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer*

Applications, *30*(1), 114–132.

Peddabachigari, S., Abraham, A., & Thomas, J. (2004). Intrusion Detection Systems Using Decision Trees and Support Vector Machines. *International Journal of Applied Science and Computations*, *11*(3), 118–134.

Powell, D., & Stroud, R. (2003). Conceptual model and architecture of MAFTIA. *Technical Report Series-University of Newcastle Upon Tyne Computing Science*.

Puri, A., & Sharma, N. (2017). A Novel Technique for Intrusion Detection System for Network Security Using Hybrid Svm-Cart. *International Journal of Engineering Development and Research (IJEDR)*, *5*(2), 155–161.

Quiroz, S., & Macías, D. (2017). Seguridad en informática: consideraciones. *Dominio de Las Ciencias*, *3*(4), 137–156.

Radford, B. J., Richardson, B. D., & Davis, S. E. (2018). Sequence Aggregation Rules for Anomaly Detection in Computer Network Traffic.

Raman, M. R. G., Somu, N., Kirthivasan, K., & Sriram, V. S. S. (2017). A Hypergraph and Arithmetic Residue-based Probabilistic Neural Network for classification in Intrusion Detection Systems. *Neural Networks*, *92*, 89–97.

Ranjan, R., & Sahoo, G. (2014). A New Clutering Approach for Anomaly Intrusion Detection. *International Journal of Data Mining & Knowledge Management Process*, *4*(2), 29–38.

Resende, P. A. A., & Drummond, A. C. (2018). Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. *Security and Privacy*, *1*(4), e36.

Reza, M., Miri, S., & Javidan, R. (2016). A Hybrid Data Mining Approach for Intrusion Detection on Imbalanced NSL-KDD Dataset. *International Journal of Advanced Computer Science and Applications*, *7*(6), 20–25.

Rieck, K., Trinius, P., Willems, C., & Holz, T. (2011). Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, *19*(4), 639–668.

Romero, M., Figueroa, G., Vera, D., Álava, J., Parrales, G., Álava, C., … Castillo, M. (2018). *Introducción a la seguridad informática y el análisis de vulnerabilidades*. 3ciencias.

Roopak, M., Yun Tian, G., & Chambers, J. (2019). Deep Learning Models for Cyber Security in IoT Networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0452–0457). IEEE.

Sahin, Y., Bulkan, S., & Duman, E. (2013). A cost-sensitive decision tree approach for fraud detection. *Expert Systems With Applications*, *40*(15), 5916–5923.

Salas, R. (2005). Redes Neuronales Artificiales, 1–7.

Salazar, R. (2016). *Sistema de detección de intrusos mediante modelado de URI*.

Salunkhe, U. R., & Mali, S. N. (2017). Security Enrichment in Intrusion Detection System Using Classifier Ensemble. *Journal of Electrical and Computer Engineering*, *2017*.

Sangkatsanee, P., Wattanapongsakorn, N., & Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, *34*(18), 2227–2235.

Saniee Abadeh, M., Habibi, J., Barzegar, Z., & Sergi, M. (2007). A parallel genetic local search algorithm for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence*, *20*(8), 1058–1069.

Sathyanarayana, A., Joty, S., Fernandez-Luque, L., Ofli, F., Srivastava, J., Elmagarmid, A., … Taheri, S. (2016). Sleep Quality Prediction From Wearable Data Using Deep Learning. *JMIR MHealth and UHealth*, *4*(4), e125.

Schultz, M. G., Eskin, E., Zadok, F., & Stolfo, S. J. (2002). Data mining methods for

detection of new malicious executables. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy* (pp. 38–49).

Scott, S. L. (2004). A Bayesian paradigm for designing intrusion detection systems. *Computational Statistics & Data Analysis*, *45*(1), 69–83.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1643–1647). IEEE.

Serrano, A., Soria, E., & Martín, J. (2010). Redes Neuronales Artificiales. *Escuela Técnica Superior de Ingeniería, Universidad de Valencia*.

Shafi, K., & Abbass, H. A. (2009). An adaptive genetic-based signature learning system for intrusion detection. *Expert Systems with Applications*, *36*(10), 12036–12043.

Shao, L., Cai, Z., Liu, L., & Lu, K. (2017). Performance evaluation of deep feature learning for RGB-D image/video classification. *Information Sciences*, *385–386*, 266–283.

Sharafaldin, I., Habibi, A., & Ghorbani, A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *ICISSP*, 108–116.

Shim, W., Kim, G., & Kim, S. (2010). A distributed sinkhole detection method using cluster analysis. *Expert Systems with Applications*, *37*(12), 8486–8491.

Shin, S., Lee, S., Kim, H., & Kim, S. (2013). Advanced probabilistic approach for network intrusion forecasting and detection. *Expert Systems with Applications*, *40*(1), 315–322.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(1), 41–50.

Shyu, M. L., Chen, S. C., Sarinnapakorn, K., & Chang, L. (2003). A Novel Anomaly

Detection Scheme Based on Principal Component Classifier. In *3rd IEEE International Conference on Data Mining* (pp. 353–365).

Siddiqui, A., & Farooqui, T. (2017). Improved Ensemble Technique based on Support Vector Machine and Neural Network for Intrusion Detection System. *International Journal Online of Science, 3*(12).

Siddiqui, M., Wang, M., & Lee, J. (2008). Detecting Internet Worms Using Data Mining Techniques. *Journal of Systemics, Cybernetics & Informatics, 6*, 48–53.

Silva, S. S. C., Silva, R. M. P., Pinto, R. C. G., & Salles, R. M. (2013). Botnets: A survey. *Computer Networks*, *57*(2), 378–403.

Singh, S., Raiwani, Y. P., & Panwar, L. (2019). Evaluation of Network Intrusion Detection with Features Selection and Machine Learning Algorithms on CICIDS-2017 Dataset. *SSRN Electronic Journal*.

Sivatha Sindhu, S. S., Geetha, S., & Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*, *39*(1), 129–141.

Smaha, S. E. (1988). Haystack: an intrusion detection system. In *[Proceedings 1988] Fourth Aerospace Computer Security Applications* (pp. 37–44). IEEE.

Srimuang, W., & Intarasothonchun, S. (2015). Classification model of network intrusion using Weighted Extreme Learning Machine. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)* (pp. 190–194). IEEE.

Stein, G., Chen, B., Wu, A. S., & Hua, K. A. (2005). Decision tree classifier for network intrusion detection with GA-based feature selection. In *Proceedings of the 43rd annual southeast regional conference on - ACM-SE 43* (Vol. 2, p. 136). New York, USA: ACM Press.

Stolfo, S., Fan, W., Lee, W., Prodromidis, A., & Chan, P. (2000). Cost-based modeling for fraud and intrusion detection: results from the jam project, in: DARPA Information Survivability Conference and Exposition. *DISCEX'00, Proceedings*, *2*(IEEE, 2000), 130–144.

Su, M.-Y. (2011). Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications*, *38*(4), 3492–3498.

Subba, B., Biswas, S., & Karmakar, S. (2016). Enhancing performance of anomaly based intrusion detection systems through dimensionality reduction using principal component analysis. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (pp. 1–6). IEEE.

Sumaiya, I., & Aswani, C. (2017). Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *Journal of King Saud University - Computer and Information Sciences*, *29*(4), 462–472.

Syarif, A. R., & Gata, W. (2017). Intrusion detection system using hybrid binary PSO and K-nearest neighborhood algorithm. In *2017 11th International Conference on Information & Communication Technology and System (ICTS)* (pp. 181–186). IEEE.

Tajbakhsh, A., Rahmati, M., & Mirzaei, A. (2009). Intrusion detection using fuzzy association rules. *Applied Soft Computing*, *9*(2), 462–469.

Thomas, J., Maszczyk, T., Sinha, N., Kluge, T., & Dauwels, J. (2017). Deep learning-based classification for brain-computer interfaces. In *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017* (pp. 234–239). IEEE.

Tian, R., Batten, L. M., & Versteeg, S. C. (2008). Function length as a tool for malware classification. In *3rd International Conference on Malicious and Unwanted Software, MALWARE 2008* (pp. 69–76).

Tian, Ronghua, Batten, L., Islam, R., & Versteeg, S. (2009). An automated classification

system based on the strings of trojan and virus families. In *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)* (pp. 23–30). IEEE.

Tong, S., Gu, H., & Yu, K. (2016). A comparative study of robustness of deep learning approaches for VAD. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5695–5699). IEEE.

Tong, X., Wang, Z., & Yu, H. (2009). A research using hybrid RBF/Elman neural networks for intrusion detection system secure model. *Computer Physics Communications*, *180*(10), 1795–1801.

Toosi, A. N., & Kahani, M. (2007). A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. *Computer Communications*, *30*(10), 2201–2212.

Toral, J. (2017). Redes Neuronales.

Tran, C., Vo, T. N., & Thinh, T. N. (2017). HA-IDS: A heterogeneous anomaly-based intrusion detection system. In *2017 4th NAFOSTED Conference on Information and Computer Science* (pp. 156–161). IEEE.

Tran, T., Cao, L., Tran, D., & Nguyen, C. (2009). Novel Intrusion Detection using Probabilistic Neural Network and Adaptive Boosting. *Journal of Computer Science*, *6*(1).

Übeyli, E. D. (2007). Implementing automated diagnostic systems for breast cancer detection. *Expert Systems with Applications*, *33*(4), 1054–1062.

Ustebay, S., Turgut, Z., & Aydin, M. A. (2018). Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)* (pp. 71–76). IEEE.

Verma, P., Anwar, S., Khan, S., & Mane, S. B. (2018). Network Intrusion Detection Using

Clustering and Gradient Boosting. In *9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1–7). IEEE.

Vijayanand, R., Devaraj, D., & Kannapiran, B. (2018). Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. *Computers & Security*, *77*, 304–314.

Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Evaluating deep learning approaches to characterize and classify malicious URL's. *Journal of Intelligent & Fuzzy Systems*, *34*(3), 1333–1343.

Wagner, C., State, R., Engel, T., & Learning, M. (2011). Machine Learning Approach for IP-Flow Record Anomaly Detection. In *International Conference on Research in Networking* (pp. 28–39). Springer, Berlin, Heidelberg.

Wagner, D., & Dean, R. (2001). Intrusion detection via static analysis. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001* (pp. 156–168). IEEE.

Wagner, David, & Soto, P. (2002). Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM conference on Computer and communications security - CCS '02* (p. 255). New York, USA: ACM Press.

Wang, G., Hao, J., Mab, J., & Huang, L. (2010). A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. *Expert Systems with Applications*, *37*(9), 6225–6232.

Wang, K., & Stolfo, S. J. (2004). Anomalous Payload-Based Network Intrusion Detection. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 203–222). Springer, Berlin, Heidelberg.

Wang, S.-S., Yan, K.-Q., Wang, S.-C., & Liu, C.-W. (2011). An Integrated Intrusion Detection System for Cluster-based Wireless Sensor Networks. *Expert Systems with Applications*, *38*(12), 15234–15243.

Wang, W., & Battiti, R. (2006). Identifying intrusions in computer networks with principal component analysis. In *First International Conference on Availability, Reliability and Security (ARES'06)* (p. 279). IEEE.

Wang, W., Guan, X., & Zhang, X. (2004). A Novel Intrusion Detection Method Based on Principle Component Analysis in Computer Security. In *International Symposium on Neural Networks* (pp. 657–662). Springer, Berlin, Heidelberg.

Wang, X., & Yiu, S. M. (2016). A multi-task learning model for malware classification with useful file access pattern from API call sequence.

Wankhede, S., & Kshirsagar, D. (2018). DoS Attack Detection Using Machine Learning and Neural Network. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (pp. 1–5). IEEE.

Weiming Hu, Wei Hu, & Maybank, S. (2008). AdaBoost-Based Algorithm for Network Intrusion Detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(2), 577–583.

Wu, H.-C., & Huang, S.-H. S. (2010). Neural networks-based detection of stepping-stone intrusion. *Expert Systems with Applications*, *37*(2), 1431–1437.

Wu, S.-Y., & Yen, E. (2009). Data mining-based intrusion detectors. *Expert Systems with Applications*, *36*(3), 5605–5612.

Wu, Y., Jiang, M., Xu, J., Zhi, D., & Xu, H. (2017). Clinical Named Entity Recognition Using Deep Learning Models. In *AMIA Annual Symposium Proceedings* (Vol. 2017, p. 1812). American Medical Informatics Association.

Xiang, C., Yong, P. C., & Meng, L. S. (2008). Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. *Pattern Recognition Letters*, *29*(7), 918–924.

Xu, B., Chen, S., Zhang, H., & Wu, T. (2017). Incremental k-NN SVM method in intrusion

detection. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 712–717). IEEE.

Yassin, W., Udzir, N., Muda, Z., & Sulaiman, M. (2013). Anomaly-Based Intrusion Detection Through K- Means Clustering and Naives Bayes Classification. In *Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013* (pp. 298–303).

Yi, Y., Wu, J., & Xu, W. (2011). Incremental SVM based on reserved set for network intrusion detection. *Expert Systems with Applications*, *38*(6), 7698–7707.

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, *5*, 21954–21961.

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative Study of CNN and RNN for Natural Language Processing.

Yulianto, A., Sukarno, P., & Suwastika, N. A. (2019). Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. *Journal of Physics: Conference Series*, *1192*(1), 012018.

Zaragoza, O. (2018). *Desarrollo de una red neuronal convolucional para el procesamiento de imágenes placentarias*. (Tesis de Ingeniería, Universidad Nacional Autónoma de México).

Zhang, Z., & Shen, H. (2005). Application of online-training SVMs for real-time intrusion detection with different considerations. *Computer Communications*, *28*(12), 1428–1442.

Zhao, S., Li, W., Zia, T., & Zomaya, A. Y. (2017). A Dimension Reduction Model and Classifier for Anomaly-Based Intrusion Detection in Internet of Things. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology*

Congress(DASC/PiCom/DataCom/CyberSciTech) (pp. 836–843). IEEE.

Zhou, Y.-Y., & Cheng, G. (2019). An Efficient Network Intrusion Detection System Based on Feature Selection and Ensemble Classifier.

Zhu, M., Ye, K., Wang, Y., & Xu, C.-Z. (2018). A Deep Learning Approach for Network Anomaly Detection Based on AMF-LSTM. In *IFIP International Conference on Network and Parallel Computing* (pp. 137–141). Springer, Cham.