

UNIVERSIDAD NACIONAL DE INGENIERÍA



Facultad de Ciencias Escuela Profesional de Computación

Curso: CC412 PROYECTO DE TESIS 1

Título del proyecto de investigación:

Arquitecturas LSTM para análisis de detección de intrusos en redes

Alumno:

Arturo Medardo Hinostroza Olivera 20191548K

Asesor:

Cesar Jesus Lara Avila

Lima, Perú - 2024

Dedicatoria

*Este trabajo está dedicado a mi madre, padre y hermanos,
por su ayuda cuando más la necesité,
por mi bienestar y mi educación.*

Agradecimiento

Agradecido con mi familia, especialmente a mi madre y a mi padre, quienes nunca han fallado como padres y me apoyaron en mis momentos más difíciles, son mi motivo para culminar mi carrera y seguir desarrollándome profesionalmente.

Agradezco a todas las personas que de una u otra manera han contribuido en mi formación, especialmente a mi grupo de amigos cercanos.

Resumen

El avance acelerado del Internet de las Cosas (IoT) ha aumentado la exposición de dispositivos y redes a ciberataques sofisticados, representando un riesgo considerable para individuos y organizaciones. Es imprescindible implementar medidas de seguridad robustas, como los sistemas de detección de intrusos (IDS), para proteger estos entornos.

Las arquitecturas de Long short-term memory (LSTM), una variante de las redes neuronales recurrentes, han demostrado ser especialmente efectivas en el análisis de tráfico de red para la detección de actividades maliciosas.

Este proyecto se enfoca en el diseño y aplicación de arquitecturas LSTM para el análisis de detección de intrusos en redes, buscando optimizar la capacidad de respuesta ante ciberamenazas y reducir los falsos positivos. Al aportar soluciones óptimas para el muestreo de ataques recurrentes, se contribuye al fortalecimiento de la protección de las redes y al avance en el uso de inteligencia artificial para la defensa proactiva.

Abstract

The accelerated advancement of the Internet of Things (IoT) has increased the exposure of devices and networks to sophisticated cyberattacks, representing a considerable risk for individuals and organizations. It is essential to implement robust security measures, such as intrusion detection systems (IDS), to protect these environments.

Long short-term memory (LSTM) architectures, a variant of recurrent neural networks, have proven to be especially effective in analyzing network traffic for the detection of malicious activities.

This project focuses on the design and application of LSTM architectures for intrusion detection analysis in networks, seeking to optimize the response capacity to cyberthreats and reduce false positives. By providing optimal solutions for sampling recurrent attacks, it contributes to strengthening network protection and advancing the use of artificial intelligence for proactive defense.

Índice de figuras

Índice de figuras

| | | |
|----|---|----|
| 1 | Tipos de arquitecturas de Deep Learning | 5 |
| 2 | Arquitectura de Red Neuronal Recurrente (RNN) | 7 |
| 3 | Arquitectura Long Short-Term Memory (LSTM) | 9 |
| 4 | Paso 1 Gated Recurrent Unit (GRU) | 12 |
| 5 | Paso 2 Gated Recurrent Unit (GRU) | 13 |
| 6 | Paso 3 Gated Recurrent Unit (GRU) | 14 |
| 7 | Conteo de datos | 22 |
| 8 | Diagrama de datos | 22 |
| 9 | Balanceo del conjunto de datos | 25 |
| 10 | Exactitud RNN | 28 |
| 11 | Exactitud LSTM | 28 |
| 12 | Exactitud GRU | 28 |
| 13 | Exactitud GRU + Atención general | 28 |
| 14 | Exactitud GRU + Atención aditiva | 28 |
| 15 | Exactitud GRU + Atención producto punto | 28 |
| 16 | Exactitud GRU + Producto punto escalado | 28 |
| 17 | Loss RNN | 29 |
| 18 | Loss LSTM | 29 |
| 19 | Loss GRU | 29 |
| 20 | Loss GRU + Atención general | 29 |
| 21 | Loss GRU + Atención aditiva | 29 |
| 22 | Loss GRU + Atención producto punto | 29 |
| 23 | Loss GRU + Producto punto escalado | 29 |
| 24 | Matriz de confusión multiclase RNN | 30 |
| 25 | Matriz de confusión multiclase LSTM | 30 |
| 26 | Matriz de confusión multiclase GRU | 30 |
| 27 | Matriz de confusión multiclase GRU + Atención general | 30 |
| 28 | Matriz de confusión multiclase GRU + Atención aditiva | 30 |
| 29 | Matriz de confusión multiclase GRU + Atención producto punto | 30 |
| 30 | Matriz de confusión multiclase GRU + Atención producto punto escalado | 30 |

| | | |
|----|--|----|
| 31 | Matriz de confusión binaria RNN | 31 |
| 32 | Matriz de confusión binaria LSTM | 31 |
| 33 | Matriz de confusión binaria GRU | 31 |
| 34 | Matriz de confusión binaria GRU + Atención general | 31 |
| 35 | Matriz de confusión binaria GRU + Atención aditiva | 31 |
| 36 | Matriz de confusión binaria GRU + Atención producto punto | 32 |
| 37 | Matriz de confusión binaria GRU + Atención producto punto escalado . . | 32 |

Abreviaturas

| | |
|-------------|--|
| IA | Inteligencia Artificial |
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| GRU | Gated Recurrent Unit |
| IDS | Intrusion Detection System |
| SIDS | Signature intrusion detection systems |
| AIDS | Anomaly-based intrusion detection system |
| UNSW | Universidad de Nueva Gales del Sur |
| DOS | Denial of Service |

Índice

| | |
|---|------------|
| Dedicatoria | I |
| Agradecimientos | II |
| Resumen | III |
| Abstract | IV |
| Índice de figuras | V |
| Índice de abreviaturas | VII |
| 1 Introducción | 1 |
| 1.1 Motivación | 1 |
| 1.2 Formulación del problema | 2 |
| 1.2.1 Pregunta principal | 2 |
| 1.2.2 Preguntas específicas | 2 |
| 1.3 Objetivos de estudio | 2 |
| 1.3.1 Objetivo general | 2 |
| 1.3.2 Objetivos específicos | 2 |
| 1.4 Estructura del proyecto | 3 |
| 2 Antecedentes y Estado del arte | 4 |
| 2.1 Sistemas de Detección de Intrusos (IDS) nociones básicas | 4 |
| 2.1.1 ¿Qué son las IDS? | 4 |
| 2.1.2 Limitaciones de los sistemas tradicionales | 4 |
| 2.2 Fundamentos del Deep Learning aplicado a la seguridad cibernética | 4 |
| 2.2.1 Arquitectura general de modelos de Deep Learning | 5 |
| 2.3 Técnica de Red Neuronal Recurrente (RNN) y recursos | 6 |
| 2.3.1 ¿Qué es una RNN? | 6 |
| 2.3.2 Arquitectura RNN | 6 |
| 2.3.3 Modelo Matemático | 7 |
| 2.4 Técnica Long Short-Term Memory (LSTM) y recursos | 8 |
| 2.4.1 ¿Qué es LSTM? | 8 |
| 2.4.2 Arquitectura LSTM | 8 |
| 2.4.3 Modelo Matemático | 9 |
| 2.5 Técnica Gated Recurrent Unit (GRU) y recursos | 11 |
| 2.5.1 ¿Qué es GRU? | 11 |
| 2.5.2 Arquitectura GRU | 12 |
| 2.5.3 Modelo matemático | 12 |

| | | |
|----------|---|-----------|
| 2.6 | Técnica Additive y recursos | 14 |
| 2.6.1 | ¿Qué es el Additive Attention? | 14 |
| 2.6.2 | Arquitectura Additive | 14 |
| 2.6.3 | Modelo matemático | 15 |
| 2.7 | Técnica General y recursos | 15 |
| 2.7.1 | ¿Qué es el General Attention? | 15 |
| 2.7.2 | Arquitectura General | 15 |
| 2.7.3 | Modelo matemático | 15 |
| 2.8 | Técnica Dot-Product y recursos | 16 |
| 2.8.1 | ¿Qué es el Dot-Product Attention? | 16 |
| 2.8.2 | Arquitectura Dot-Product | 16 |
| 2.8.3 | Modelo matemático | 16 |
| 2.9 | Técnica Scaled Dot-Product y recursos | 16 |
| 2.9.1 | ¿Qué es el Scaled Dot-Product Attention? | 16 |
| 2.9.2 | Arquitectura Scaled Dot-Product | 16 |
| 2.9.3 | Modelo matemático | 17 |
| 2.10 | Técnica SMOTE y recursos | 17 |
| 2.10.1 | ¿Qué es SMOTE? | 17 |
| 2.10.2 | Arquitectura SMOTE | 17 |
| 2.10.3 | Modelo matemático | 17 |
| 2.11 | Técnica Borderline-SMOTE y recursos | 18 |
| 2.11.1 | ¿Qué es Borderline-SMOTE? | 18 |
| 2.11.2 | Arquitectura Borderline-SMOTE | 18 |
| 2.11.3 | Modelo matemático | 18 |
| 2.12 | Técnica SMOTE-ENC y recursos | 18 |
| 2.12.1 | ¿Qué es SMOTE-ENC? | 18 |
| 2.12.2 | Arquitectura SMOTE-ENC | 18 |
| 2.12.3 | Modelo matemático | 19 |
| 2.13 | Técnica ADASYN y recursos | 19 |
| 2.13.1 | ¿Qué es ADASYN? | 19 |
| 2.13.2 | Arquitectura ADASYN | 19 |
| 2.13.3 | Modelo matemático | 19 |
| 2.13.4 | Informe sobre el Dataset UNSW-NB15 | 20 |
| 3 | Desarrollo del trabajo de investigación | 22 |
| 3.1 | Tratamiento de Datos | 22 |
| 3.2 | Análisis de desequilibrio en los datos | 22 |
| 3.2.1 | Distribución de la etiqueta label (Normal vs. Ataque) | 23 |
| 3.2.2 | Diferencias en las cantidades | 23 |

| | |
|---|-----------|
| 3.3 Definición de modelos utilizados | 25 |
| 3.4 Datos resultantes del entrenamiento | 27 |
| 4 Análisis, discusiones de resultados y conclusiones | 33 |
| 4.1 Análisis y discusiones de resultados | 33 |
| 4.2 Conclusiones | 34 |

Capítulo 1

1. INTRODUCCIÓN

1.1. Motivación

En la era digital, la inteligencia artificial (IA) representa tanto una oportunidad como un riesgo dentro del campo de la ciberseguridad. Con el desarrollo de esta tecnología, los ataques cibernéticos se han vuelto más avanzados y frecuentes. La IA facilita la automatización de actividades maliciosas como el malware, el phishing y el robo de información, proporcionando a los ciberdelincuentes herramientas más eficaces para identificar y explotar debilidades en los sistemas. Estos ataques comprometen datos sensibles de millones de usuarios y se han ido incrementando a través de estos últimos años. [1]

En Perú, también son un riesgo creciente. La desarrolladora de antivirus ESET registró un 40 % más de malware malicioso en nuestro país entre enero y agosto de este año. En tanto, la firma de seguridad informática Kaspersky advirtió que en el caso de las pequeñas y medianas empresas (pymes) peruanas los ataques llegaron a 9,6 millones de octubre de 2022 al mismo mes de 2023, lo que supone un 3 % más que en igual período del año anterior. [2]

A medida que más organizaciones dependen de la digitalización para sus operaciones diarias, la integridad, confidencialidad y disponibilidad de los datos se vuelven críticas. Una violación puede llevar a pérdidas irreparables en la reputación de una empresa y litigios costosos. Por lo que es necesario implementar un sistema de detección de ataques o intrusos para mitigar los riesgos.

Tradicionalmente, los sistemas de detección de intrusos (IDS) han estado basados en la detección de firmas para identificar amenazas conocidas. Sin embargo, este enfoque presenta limitaciones ante nuevas variantes de ataques, que evolucionan y se adaptan para eludir los controles de seguridad tradicionales. Ante este desafío, las redes neuronales emergen como una solución prometedora debido a su capacidad para aprender de grandes volúmenes de datos. [3]

En este contexto, surge la necesidad de desarrollar nuevos modelos que sean más eficientes y robustos para enfrentar las amenazas emergentes. Por tal motivo, presento la resolución de mi proyecto de investigación titulado ‘Arquitecturas LSTM para análisis de detección de intrusos en redes’.

1.2. Formulación del problema

1.2.1. Pregunta principal

¿Las arquitecturas tradicionales de redes neuronales (LSTM, GRU y RNN) comparadas con una arquitectura GRU mejorada con mecanismos de atención pueden mejorar la detección de intrusos en redes en términos de precisión y reducción de falsas alarmas?

1.2.2. Preguntas específicas

¿Cuál es el impacto de los diferentes mecanismos de atención en el desempeño de una arquitectura GRU para la detección de intrusos en redes?

¿Cómo se compara el rendimiento de una arquitectura GRU con mecanismos de atención en términos de precisión, tasa de falsas alarmas y capacidad de detección de ataques?

1.3. Objetivos de estudio

1.3.1. Objetivo general

Analizar y comparar la efectividad de modelos (en relación de precisión) de detección de intrusos en redes utilizando arquitecturas tradicionales de redes neuronales (LSTM, GRU y RNN) como base, y posteriormente evaluar el desempeño de una arquitectura GRU mejorada con mecanismos de atención para mejorar la precisión y capacidad de detección de intrusos.

1.3.2. Objetivos específicos

Implementar y analizar el desempeño de una arquitectura GRU con distintos mecanismos de atención, visualizando su impacto en la precisión y capacidad de detección de intrusos.

Proponer una arquitectura GRU con mecanismos de atención y visualizar su rendimiento en comparación con las arquitecturas tradicionales.

Comparación de rendimientos en términos de precisión, tasa de falsas alarmas y capacidad de detección de ataques.

1.4. Estructura del proyecto

Para la orientación general del lector se detalla la siguiente estructura; de este modo, se facilita la búsqueda de información por capítulos.

- **Capítulo 1: INTRODUCCIÓN**

En este capítulo se exponen la razón y los motivos generales por el cual se realizó el desarrollo del presente trabajo, asimismo expone los objetivos y preguntas a responder del proyecto.

- **Capítulo 2: ANTECEDENTES Y ESTADO DEL ARTE**

En este capítulo se expone definiciones y conceptos importantes para la fácil comprensión del proyecto a lo largo de su desarrollo, asimismo expresa las arquitecturas a utilizar y la definición de los mecanismos de atención a utilizar.

- **Capítulo 3: DESARROLLO DEL TRABAJO DE INVESTIGACIÓN**

En este capítulo se expone el diagrama de flujo del trabajo de investigación en la parte de implementación, desde la parte inicial de los tratamiento de datos hasta la culminación y comparativa de las pautas del código implementado.

- **Capítulo 4: ANÁLISIS, DISCUSIONES DE RESULTADOS Y CONCLUSIONES**

Se expone las ideas finales derivadas de los resultados de la implementación realizada en en capítulo 3.

Capítulo 2

2. ANTECEDENTES Y ESTADO DEL ARTE

2.1. Sistemas de Detección de Intrusos (IDS) nociones básicas

2.1.1. ¿Qué son las IDS?

Un sistema de detección de intrusos (IDS) es una aplicación la cual adquiere información sobre un sistema para realizar un diagnóstico sobre el estado de seguridad del mismo. El objetivo es descubrir anomalías en el sistema, ya sea violaciones de seguridad, intentos de violación o apertura de vulnerabilidades que podrían conducir a posibles infracciones. [4]

2.1.2. Limitaciones de los sistemas tradicionales

- **Sistemas de detección de intrusos basados en firmas (SIDS):**

Se utilizan métodos de comparación para encontrar una intrusión anterior, es decir, cuando una firma de intrusión coincide con la firma de una intrusión anterior que ya existe en la base de datos de firmas se activa una señal de alarma. Este tipo de detección es ineficiente debido a que no puede detectar ataques del día cero, es decir, intrusiones los cuales no se encuentren en la firma anterior. Debido a esta desventaja y al crecimiento acelerado de recursos tecnológicos para realizar variantes polimórficas de los malware al realizar ciberataques, se ha vuelto menos factible utilizar SIDS. [5]

- **Sistema de detección de intrusiones basado en anomalías (AIDS):**

Los AIDS superan las limitaciones de los SIDS, sin embargo, aún presenta limitaciones como: No pueden manejar paquetes encriptados, lo que significa que los ataques que utilizan cifrado pueden pasar desapercibidos. A menudo generan falsos positivos debido a su enfoque basado en desviaciones por lo que Las alertas no siempre se clasifican adecuadamente. [5]

2.2. Fundamentos del Deep Learning aplicado a la seguridad cibernética

El Deep Learning, también conocido como aprendizaje profundo, es una rama de la inteligencia artificial (IA) Se basa en el uso de redes neuronales artificiales de múltiples capas para aprender y realizar tareas que antes eran difíciles o imposibles de lograr con algoritmos tradicionales. [6]

2.2.1. Arquitectura general de modelos de Deep Learning

Como se muestra en la Figura 1, existen varios tipos de arquitecturas en el ámbito del Deep Learning que se pueden aplicar para mejorar la eficiencia de los modelos predictivos. Aunque son diferentes, todas ellas comparten similitud de un modelo estructural en los que se encuentran los siguientes elementos.

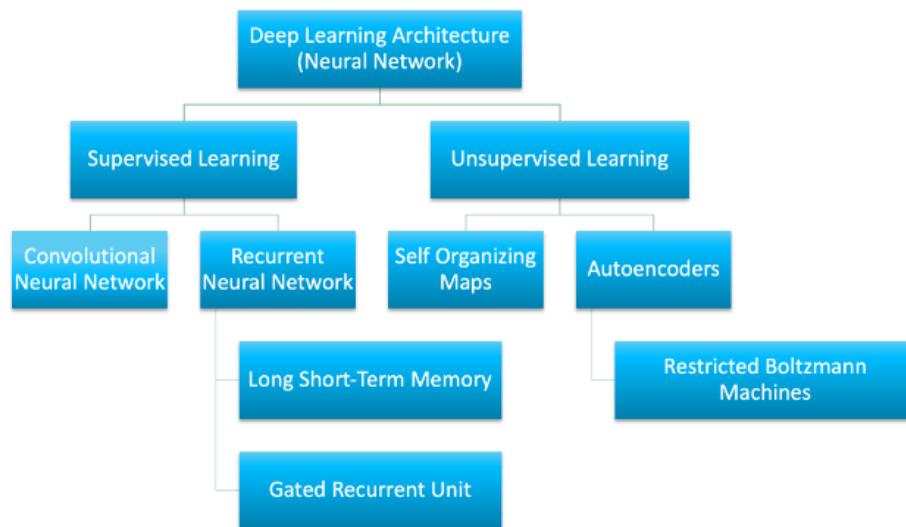


Figura 1: Tipos de arquitecturas de Deep Learning [7]

- **Neurona:** Las neuronas artificiales son una unidad básica de procesamiento en una red neuronal. [8]
- **Red neuronal:** Conjunto de neuronas que transmiten datos entre sí. [8]
- **Capa de Entrada:** La información a procesar entra en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y los pasan a la siguiente capa. [9]
- **Capas Ocultas:** Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas, cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa. [9]
- **Capa de Salida:** Se trata de la capa que realiza alguna conclusión o toma una decisión y que aporta los datos finales del proceso. Puede estar conformada por una o varias capas. [10]

- **Funciones de Activación:** Estas funciones introducen no linealidad en el modelo, lo que le permite aprender relaciones complejas entre input (datos de entrada) y output (entradas de salida). [11]
- **Mecanismo de Aprendizaje:** Se refiere a cómo las redes neuronales adaptan sus parámetros (pesos y sesgos) durante el entrenamiento para mejorar su rendimiento en una tarea específica. [12] Se utilizan mecanismos como: Backpropagation, Momentum, etc.
- **Regularización:** Modificación realizada a un algoritmo para reducir el error en un conjunto de datos de prueba sin afectar significativamente al conjunto de datos de entrenamiento. [13] Se utilizan técnicas de regularización como: Dropout, Early Stopping, etc.

2.3. Técnica de Red Neuronal Recurrente (RNN) y recursos

2.3.1. ¿Qué es una RNN?

Las Redes Neuronales Recurrentes (RNN) son un tipo de arquitectura de red neuronal diseñada para procesar datos secuenciales, permitiendo que la información fluya de un paso de tiempo a otro. A diferencia de las redes neuronales tradicionales, las RNN tienen conexiones recurrentes que les permiten retener información sobre pasos de tiempo previos. Esto las hace particularmente útiles en tareas como el análisis de series temporales, procesamiento de lenguaje natural y otras aplicaciones que requieren manejar dependencias temporales. Se tiene como estructura general lo siguiente:

2.3.2. Arquitectura RNN

El modelo RNN utilizado se encuentra completamente conectado, donde la salida debe realimentarse como nueva entrada. El modelo RNN se divide en los siguientes componentes principales [14] [15]:

- **Entrada:** La entrada (x_t) representa los datos en el paso de tiempo t . Es un vector de características que se utiliza junto con el estado oculto anterior para calcular el estado oculto actual.
- **Estado oculto:** El estado oculto (h_t) almacena información acumulada de pasos de tiempo previos. Es calculado como una combinación de la entrada actual (x_t) y

el estado oculto anterior (h_{t-1}).

- **Salida:** La salida (o_t) es generada en cada paso de tiempo en función del estado oculto actual (h_t). En aplicaciones como clasificación de secuencias, esta salida se utiliza para tomar decisiones basadas en la información procesada hasta el momento.

2.3.3. Modelo Matemático

Se tiene como referencia la Figura 2. Las ecuaciones del modelo RNN son las siguientes:

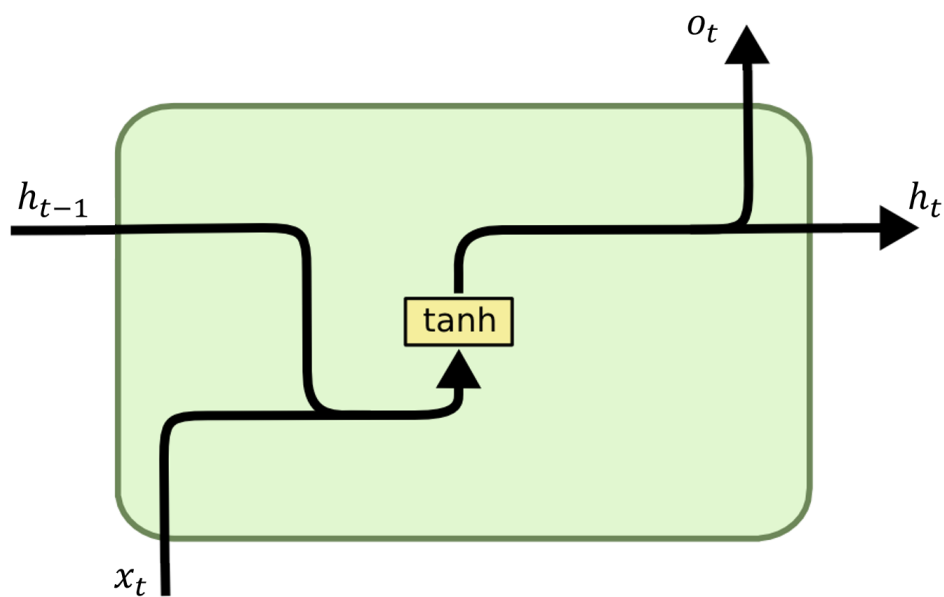


Figura 2: Arquitectura de Red Neuronal Recurrente (RNN) [16]

Estado Oculto:

$$h_t = \tanh(W_h \cdot x_t + U_h \cdot h_{t-1} + b_h) \quad (1)$$

Donde:

- h_t : Estado oculto en el paso de tiempo t .
- x_t : Entrada en el paso de tiempo t .
- h_{t-1} : Estado oculto del paso de tiempo anterior.
- W_h : Matriz de pesos aplicada a la entrada.
- U_h : Matriz de pesos aplicada al estado oculto anterior.
- b_h : Vector de sesgos.
- \tanh : Función de activación tangente hiperbólica.

Salida:

$$o_t = \sigma(W_o \cdot h_t + b_o) \quad (2)$$

Donde:

- o_t : Salida en el paso de tiempo t .
- h_t : Estado oculto en el paso de tiempo t .
- W_o : Matriz de pesos para la salida.
- b_o : Vector de sesgos.
- σ : Función de activación `softmax` para clasificación.

2.4. Técnica Long Short-Term Memory (LSTM) y recursos

2.4.1. ¿Qué es LSTM?

Long Short-Term Memory (LSTM) son un tipo de arquitectura de red neuronal recurrente (RNN) diseñada para abordar las limitaciones de las RNN tradicionales (desvanecimiento o explosión de la gradiente). Los LSTM son particularmente efectivos para manejar datos secuenciales con dependencias a largo plazo. [17] Se tiene como estructura general lo siguiente:

2.4.2. Arquitectura LSTM

El modelo LSTM se divide en Estado de la celda, Puertas y Estado oculto. [18] [19] Se tiene la siguiente representación visual del modelo en la Figura 3.

■ Estado de la celda:

El componente principal de un LSTM es la celda de memoria. El estado de la célula actúa como una cinta transportadora, permitiendo que la información fluya a través de pasos de tiempo sin mucha modificación. Puede almacenar y recuperar información en secuencias largas.

■ Puertas:

Los LSTM utilizan tres puertas para controlar el flujo de información:

- **Puerta de olvido:** Decide cuánto de la información anterior debe ser olvidada.
- **Puerta de entrada:** Decide qué nueva información almacenar en el estado de la celda.
- **Puerta de salida:** Decide cuánto del estado de la celda actual debe ser usado para generar el estado oculto.

■ Estado oculto (h):

El estado oculto (h) es la salida de la celda LSTM. Transporta información de pasos de tiempo anteriores e influye en la predicción actual.

2.4.3. Modelo Matemático

Se tiene como referencia la Figura 3. Las ecuaciones de la celda LSTM son las siguientes: [20]

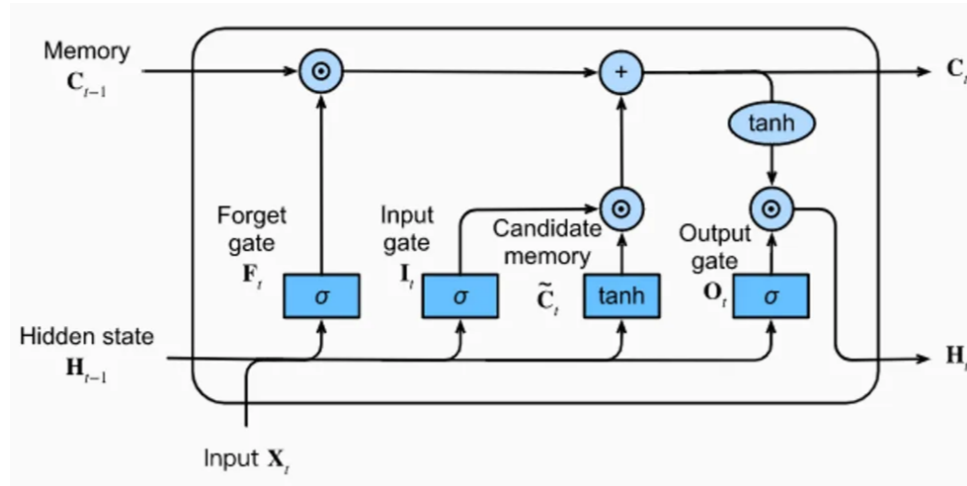


Figura 3: Arquitectura Long Short-Term Memory (LSTM) [21]

W_f representa la matriz de pesos de la puerta de olvido. Es un conjunto de coeficientes que la red aprende durante el proceso de entrenamiento. Estos pesos son utilizados para determinar cuánta información pasada se debe mantener o descartar a través de la puerta de olvido.

Cuando la red procesa una nueva entrada en cada paso de tiempo, utiliza W_f , junto con el estado anterior h_{t-1} y la entrada actual x_t , multiplicando estos valores y sumándolos con un sesgo b_f . Esta combinación luego pasa a través de una función de activación (en este caso, la función sigmoide σ), que decide qué parte de la información antigua en la celda de memoria C_{t-1} es relevante y debería retenerse para futuros cálculos. El resultado, f_t , es la salida de la puerta de olvido que se usa para multiplicar el estado de la celda anterior y así decidir cuánta información antigua se olvida.

Puerta de Olvido:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

f_t representa la salida de la puerta de olvido en el estado t.

W_f y b_f son los pesos y sesgos que se pueden aprender para la puerta de olvido f.

σ es la función de activación sigmoide.

Paso 2: El siguiente paso es tomar una decisión sobre qué nueva información se va a almacenar en el estado de la celda. Esto consta de dos partes: La primera, una capa sigmoidea llamada capa de puerta de entrada que decide qué valores se actualizarán; la segunda, una capa tanh crea un vector de nuevos valores contendientes, que podrían agregarse al estado. El siguiente paso es combinar estos dos para crear una actualización del estado.

Puerta de Entrada:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

i_t representa la salida de la puerta de entrada.

Estado de Celda Candidato:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (5)$$

\tilde{C}_t es el estado de celda candidato.

Paso 3: Ahora es el momento de actualizar el antiguo estado de la celda $C_{(t-1)}$ al nuevo estado de la celda. Los pasos anteriores ya decidieron qué hacer, en realidad solo falta hacerlo. Primero, el viejo estado se multiplica, olvidando las cosas que decidimos olvidar anteriores. Segundo: luego, agregue al estado de la celda. Estos son los nuevos valores candidatos, escalados según la proporción que decidió actualizar cada valor de estado.

Actualización del Estado de Celda:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (6)$$

C_t es el estado de celda actualizado.

Paso 4: La etapa final es la etapa de salida. Esta salida será compatible con el estado de nuestra celda, pero será una versión filtrada. Primero, ejecute una capa sigmoidea que decida qué partes del estado de la celda se generarán. Luego, pase el estado de la celda a través de tanh (para hacer que los valores estén entre -1 y 1) y multiplíquelo por la salida de la puerta sigmoidea, de modo que solo genere las partes decididas.

Puerta de Salida:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

o_t representa la salida de la puerta de salida.

Estado Oculto:

$$h_t = o_t \cdot \tanh(C_t) \quad (8)$$

h_t es el estado oculto.

Propósito de la función sigmoideal σ : Su propósito es controlar la cantidad de información que pasa a través de estas puertas, permitiendo que el modelo regule el flujo de información a lo largo del tiempo. Las puertas de entrada, olvido y salida en una LSTM utilizan la función sigmoideal para decidir cuánto de la información pasada o nueva debe ser retenida o desechada. Dado que la salida de la función sigmoideal está entre 0 y 1, esto permite que la red controle de manera suave la cantidad de información que se permite pasar a través de estas puertas.

Propósito de la función \tanh : Tanh es usada dentro de las LSTMs para regular la naturaleza de los valores de salida del estado de la celda, manteniéndolos en un rango de -1 a 1. Esto ayuda a mantener la estabilidad numérica de la red, evitando que los valores se vuelvan demasiado grandes o pequeños durante el entrenamiento. A diferencia de la sigmoidea, tanh tiene un rango de salida simétrico que centra los datos, lo cual mejora la eficiencia del aprendizaje al modelar entradas que tienen una naturaleza positiva y negativa.

La combinación de la función sigmoideal y tanh permite una actualización controlada del estado de la celda C_t , lo cual es crucial para mantener la estabilidad del modelo a través del tiempo y evitar problemas de desvanecimiento y explosión de gradientes.

2.5. Técnica Gated Recurrent Unit (GRU) y recursos

2.5.1. ¿Qué es GRU?

Gated Recurrent Unit (GRU) es un tipo de arquitectura de red neuronal recurrente (RNN) diseñada con afluencia de la arquitectura LSTM (Long Short-Term Memory). Al igual que LSTM, está orientada a resolver los problemas de las RNN tradicionales, como el desvanecimiento o explosión del gradiente. Sin embargo, la principal diferencia es que las GRU tienen una estructura más simple, lo que las hace computacionalmente más eficientes. Las GRU se especializan en manejar secuencias de datos con dependencias a largo plazo, eliminando la necesidad de una celda de memoria separada, ya que unifican la memoria y el estado oculto en un solo vector. Se tiene como estructura general lo siguiente. [22] [23]

2.5.2. Arquitectura GRU

El modelo GRU se divide en Unidad de recurrencia, Puertas y Estado oculto. [24]

- **Unidad de recurrencia:** Una GRU tiene una unidad de recurrencia que procesa la entrada actual y la salida anterior para generar una nueva salida y un estado oculto.
- **Actualización y Reinicio (Update and Reset Gates):** Las GRU utilizan dos puertas principales para controlar el flujo de información:
 - **Puerta de Actualización (Update Gate):** Decide cuánta información de la entrada y el estado anterior debe mantenerse en el estado oculto actual.
 - **Puerta de Reinicio (Reset Gate):** Controla qué información olvidar del estado anterior antes de considerar la nueva entrada.
- **Estado Oculto (Hidden State):** Es la salida de la unidad recurrente y contiene la información relevante para el siguiente paso de tiempo.

2.5.3. Modelo matemático

Las ecuaciones para la arquitectura GRU son las siguientes: [25]

Paso 1: Puerta de Reinicio y Puerta de Actualización

En este paso, se calculan las puertas de reinicio (R_t) y de actualización (Z_t) a partir de las entradas actuales y el estado oculto anterior. (Figura 4)

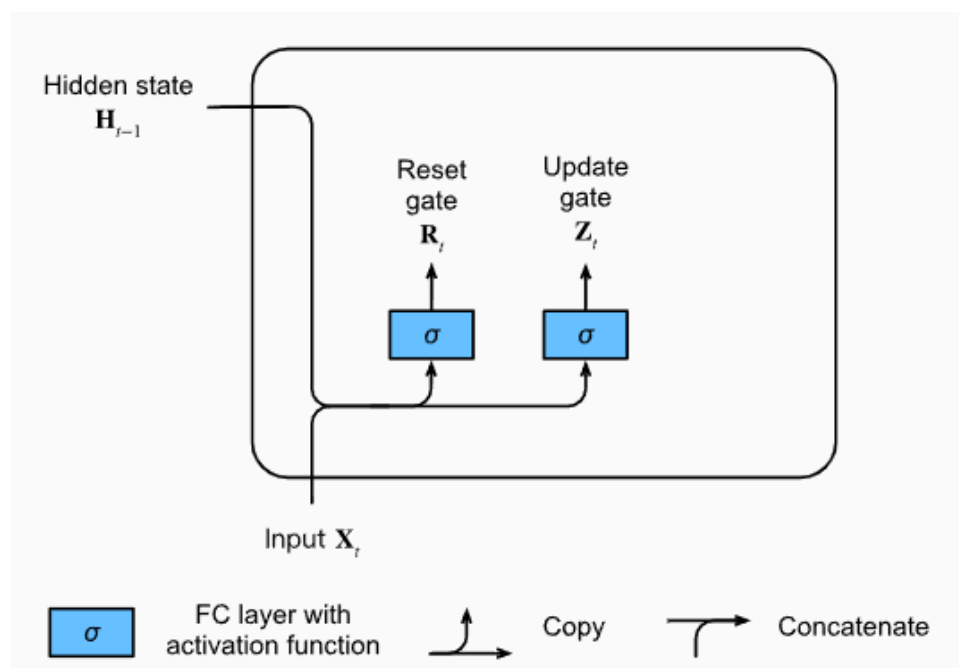


Figura 4: Paso 1 Gated Recurrent Unit (GRU) [25]

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \quad (9)$$

$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \quad (10)$$

Donde:

- X_t : Entrada en el tiempo t .
- H_{t-1} : Estado oculto en el tiempo $t - 1$.
- W_{xr}, W_{xz} : Matrices de pesos asociadas a la entrada X_t .
- W_{hr}, W_{hz} : Matrices de pesos asociadas al estado oculto previo H_{t-1} .
- b_r, b_z : Vectores de sesgo para las puertas de reinicio y actualización.
- σ : Función sigmoïdal.

Paso 2: Estado Oculto Candidato

A continuación, integramos la puerta de reinicio R_t para calcular el estado oculto candidato \tilde{H}_t . (Figura 5)

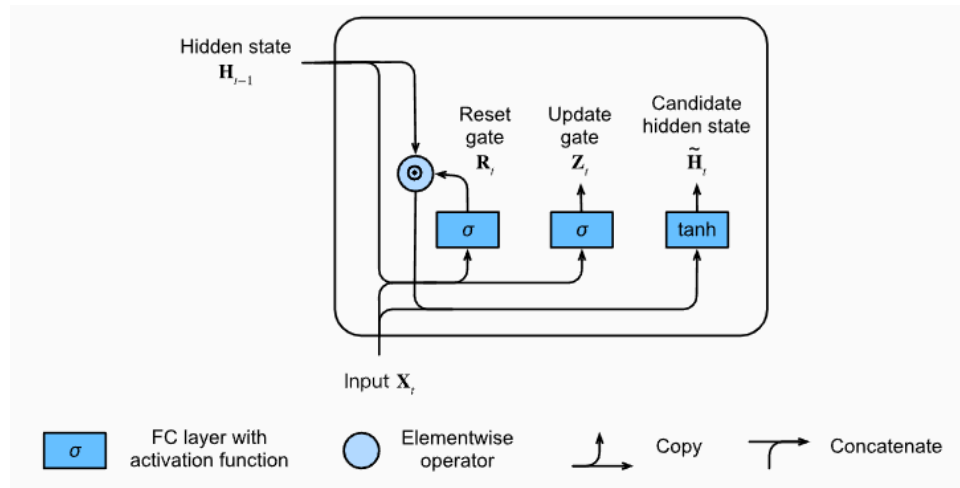


Figura 5: Paso 2 Gated Recurrent Unit (GRU) [25]

$$\tilde{H}_t = \tanh(X_t W_{zh} + (R_t \odot H_{t-1}) W_{hh} + b_h) \quad (11)$$

Donde:

- W_{zh}, W_{hh} : Matrices de pesos.
- b_h : Vector de sesgo.
- \odot : Producto de Hadamard (multiplicación elemento a elemento).
- \tanh : Función tangente hiperbólica.

Paso 3: Actualización del Estado Oculto

Finalmente, el estado oculto H_t se actualiza con la siguiente fórmula, utilizando la puerta de actualización Z_t para ponderar la influencia del estado oculto anterior y el estado candidato. (Figura 6)

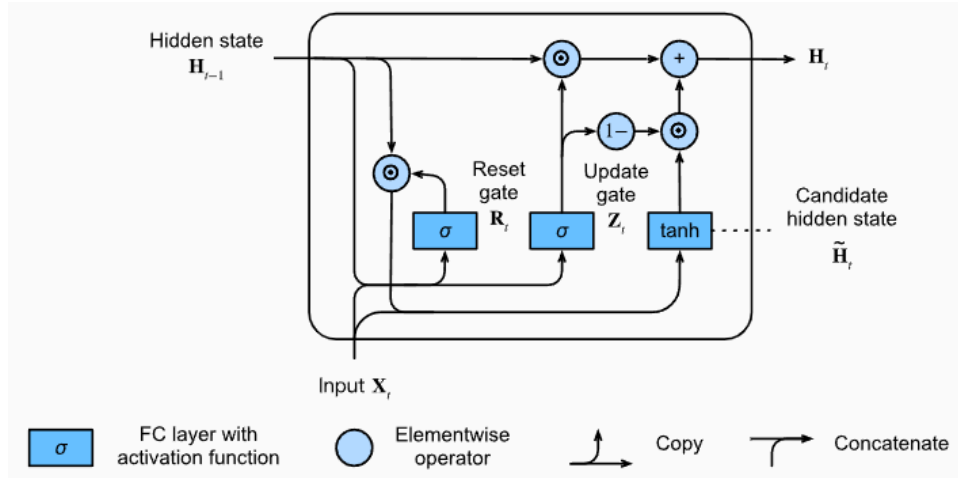


Figura 6: Paso 3 Gated Recurrent Unit (GRU) [25]

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t \quad (12)$$

Donde:

- H_t : Estado oculto en el tiempo t .
- Z_t : Puerta de actualización.
- \tilde{H}_t : Estado oculto candidato.

2.6. Técnica Additive y recursos

2.6.1. ¿Qué es el Additive Attention?

Este mecanismo utiliza una combinación lineal de las representaciones ocultas de la secuencia fuente y el estado oculto actual del decodificador. Luego, aplica una función de activación (generalmente \tanh) para calcular la relevancia de cada elemento en la secuencia de entrada. Este proceso permite al modelo enfocarse en diferentes partes de la secuencia al generar la salida. [26]

2.6.2. Arquitectura Additive

El modelo Additive emplea una capa de red neuronal para combinar la representación de la entrada y el estado oculto del decodificador, generando un vector de atención que determina las partes relevantes de la secuencia fuente en cada paso de decodificación.

- **Cálculo del puntaje:** Cada puntaje de atención se obtiene aplicando una función de activación sobre la combinación lineal del vector de la secuencia fuente y el estado actual del decodificador.
- **Matriz de pesos:** Los pesos de atención se calculan aplicando softmax al puntaje obtenido, generando un enfoque suave sobre la secuencia fuente.

2.6.3. Modelo matemático

$$score(s_t, h_i) = v_a^T \tanh(W_a[s_{t-1}, h_i]) \quad (13)$$

Donde:

- s_t : Estado actual del decodificador.
- h_i : Representación de la entrada.
- v_a y W_a : Parámetros aprendidos.

2.7. Técnica General y recursos

2.7.1. ¿Qué es el General Attention?

Adapta la idea de atención multiplicativa. A diferencia del Additive Attention, el General Attention utiliza una matriz de pesos aprendida para adaptar el estado de cada entrada y calcular el puntaje de atención. [27]

2.7.2. Arquitectura General

En el General Attention, el puntaje se calcula multiplicando el estado del decodificador con una versión ponderada de cada estado de la secuencia fuente.

- **Matriz de pesos:** Utiliza una matriz de pesos aprendida que transforma el estado de cada entrada antes de calcular el producto punto con el estado del decodificador.
- **Softmax:** Al aplicar softmax a los puntajes obtenidos, se obtienen los pesos de atención, que indican las áreas de la secuencia fuente en las que el decodificador debe enfocarse.

2.7.3. Modelo matemático

$$score(s_t, h_i) = s_t^T W_a h_i \quad (14)$$

Donde:

- s_t : Estado actual del decodificador.
- h_i : Representación de la entrada.
- W_a : Matriz de pesos aprendida.

2.8. Técnica Dot-Product y recursos

2.8.1. ¿Qué es el Dot-Product Attention?

En este mecanismo de atención el puntaje se calcula directamente a partir del producto punto entre el estado del decodificador y el de cada posición de la entrada sin aplicar transformaciones adicionales. [27]

2.8.2. Arquitectura Dot-Product

Este mecanismo es eficiente en términos computacionales, ya que evita transformaciones adicionales al calcular el puntaje de atención. Sin embargo, puede ser menos efectivo para entradas de mayor dimensionalidad debido a la posible saturación del softmax.

- **Producto punto:** El cálculo del puntaje se realiza directamente mediante el producto punto entre el estado del decodificador y el vector de la entrada.
- **Peso de atención:** El peso de atención se obtiene aplicando softmax sobre los puntajes, priorizando las entradas más relevantes.

2.8.3. Modelo matemático

$$score(s_t, h_i) = s_t^T h_i \quad (15)$$

Donde:

- s_t : Estado actual del decodificador.
- h_i : Representación de la entrada.

2.9. Técnica Scaled Dot-Product y recursos

2.9.1. ¿Qué es el Scaled Dot-Product Attention?

Es una variación del Dot-Product Attention que escala el resultado del producto punto para evitar problemas de saturación del softmax en casos de dimensiones altas. Esta técnica es utilizada principalmente en modelos de arquitectura basada en atención como el Transformer. [28]

2.9.2. Arquitectura Scaled Dot-Product

Esta arquitectura introduce un factor de escala, que evita que el producto punto alcance magnitudes demasiado altas cuando la dimensión de los vectores de entrada es grande.

- **Producto punto escalado:** Divide el producto punto por la raíz cuadrada de la dimensión de las entradas, normalizando los valores y manteniendo la estabilidad de los gradientes.
- **Softmax:** Aplica softmax para calcular los pesos de atención, facilitando el enfoque sobre las entradas más relevantes.

2.9.3. Modelo matemático

$$score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}} \quad (16)$$

Donde:

- s_t : Estado actual del decodificador.
- h_i : Representación de la entrada.
- n : Dimensión de los vectores de entrada.

2.10. Técnica SMOTE y recursos

2.10.1. ¿Qué es SMOTE?

SMOTE (Synthetic Minority Over-sampling Technique) es una técnica de sobremuestreo sintético que genera ejemplos sintéticos de la clase minoritaria para balancear la distribución de clases en conjuntos de datos desbalanceados. Esta técnica mejora el rendimiento de clasificación en problemas donde hay desbalance de clases. [29]

2.10.2. Arquitectura SMOTE

SMOTE genera ejemplos sintéticos de la clase minoritaria interpolando entre las instancias de dicha clase. Para cada instancia de la clase minoritaria, se seleccionan k vecinos más cercanos, y se crean nuevas instancias en puntos de interpolación entre la instancia original y sus vecinos.

2.10.3. Modelo matemático

$$\text{instancia_generada} = x_i + (x_{zi} - x_i) \times \lambda \quad (17)$$

Donde:

- x_i : instancia original de la clase minoritaria.
- x_{zi} : uno de los vecinos cercanos de x_i .
- λ : valor aleatorio entre 0 y 1.

2.11. Técnica Borderline-SMOTE y recursos

2.11.1. ¿Qué es Borderline-SMOTE?

Borderline-SMOTE es una variación de SMOTE que genera ejemplos sintéticos únicamente para las instancias de la clase minoritaria que están cerca del límite de decisión entre clases. Esto mejora la precisión en la clasificación de instancias difíciles de la clase minoritaria. [30]

2.11.2. Arquitectura Borderline-SMOTE

En Borderline-SMOTE, se identifican las instancias de la clase minoritaria que están cerca de la frontera de decisión. Estas instancias se consideran en 'peligro' y se priorizan para la generación de datos sintéticos, aumentando su representación en el límite de decisión.

2.11.3. Modelo matemático

Borderline-SMOTE aplica la misma fórmula de SMOTE para generar muestras sintéticas, pero con la condición de que las instancias seleccionadas deben estar cerca de la frontera de decisión entre la clase minoritaria y la clase mayoritaria:

$$\text{instancia_generada} = x_i + (x_{zi} - x_i) \times \lambda \quad (18)$$

Donde:

- x_i : instancia de la clase minoritaria en peligro.
- x_{zi} : vecino cercano de x_i en la clase minoritaria.
- λ : valor aleatorio entre 0 y 1.

2.12. Técnica SMOTE-ENC y recursos

2.12.1. ¿Qué es SMOTE-ENC?

SMOTE-ENC es una variante de SMOTE diseñada para conjuntos de datos que contienen tanto características continuas como categóricas. Esta técnica permite el sobremuestreo de datos combinados utilizando interpolación para variables continuas y asignación de categorías para variables categóricas. [31]

2.12.2. Arquitectura SMOTE-ENC

En SMOTE-ENC, las distancias entre instancias se calculan considerando solo las características continuas, mientras que las variables categóricas son asignadas utilizando el valor de categoría más frecuente de los vecinos cercanos seleccionados.

2.12.3. Modelo matemático

SMOTE-ENC aplica diferentes fórmulas para generar muestras sintéticas dependiendo del tipo de variable:

Para variables continuas:

$$\text{instancia_generada}_{\text{cont}} = x_i + (x_{zi} - x_i) \times \lambda \quad (19)$$

Para variables categóricas:

$$\text{instancia_generada}_{\text{cat}} = \text{modo}(\{x_i, x_{zi1}, x_{zi2}, \dots, x_{zik}\}) \quad (20)$$

Donde:

- x_i : instancia original de la clase minoritaria.
- x_{zi} : vecino cercano de x_i .
- λ : valor aleatorio entre 0 y 1.
- modo: función que selecciona la categoría más frecuente entre los vecinos.

2.13. Técnica ADASYN y recursos

2.13.1. ¿Qué es ADASYN?

ADASYN (Adaptive Synthetic Sampling) es una técnica de sobremuestreo adaptativo que genera ejemplos sintéticos de la clase minoritaria, priorizando las instancias que son difíciles de clasificar. Esto ayuda a reducir el sesgo introducido por el desbalance de clases al ajustar el límite de decisión hacia las instancias difíciles. [32]

2.13.2. Arquitectura ADASYN

ADASYN utiliza un criterio de distribución ponderada para generar más ejemplos sintéticos en las regiones donde las instancias de la clase minoritaria están más dispersas, mejorando la representatividad de las instancias difíciles.

2.13.3. Modelo matemático

ADASYN aplica la siguiente fórmula para generar muestras sintéticas, priorizando las instancias de la clase minoritaria que son difíciles de clasificar (determinadas por una métrica de densidad local):

$$\text{instancia_generada} = x_i + (x_{zi} - x_i) \times \lambda \quad (21)$$

Donde:

- x_i : instancia de la clase minoritaria.
- x_{zi} : vecino cercano de x_i .
- λ : valor aleatorio entre 0 y 1.

2.13.4. Informe sobre el Dataset UNSW-NB15

Introducción El dataset UNSW-NB15 es uno de los conjuntos de datos más utilizados en la investigación y desarrollo de sistemas de detección de intrusos (IDS). Fue creado por el Australian Centre for Cyber Security (ACCS) para proporcionar un conjunto de datos actualizado y comprensivo que incluye un amplio rango de actividades normales y maliciosas dentro de una red. El dataset UNSW-NB15 fue generado usando un entorno de red virtual llamado IXIA PerfectStorm para capturar tráfico real de red. Este entorno permitió la creación de tráfico normal y malicioso, proporcionando una base sólida para la evaluación de algoritmos de detección de intrusos. El tráfico de red capturado incluye varios tipos de ataques y actividades normales. [33]

Características del Dataset El dataset UNSW-NB15 incluye un total de 49 características para cada flujo de red registrado. Estas características se dividen en diferentes categorías que representan varios aspectos del tráfico de red. Aquí se detallan algunas de las características más relevantes:

- **Características de Flujo (Flow Features)**

- `srcip`, `sport`, `dstip`, `dport`: Dirección IP y puertos de origen y destino.
- `proto`: Protocolo utilizado (por ejemplo, TCP, UDP).
- `state`: Estado de la conexión.

- **Características de Tiempo (Time Features)**

- `dur`: Duración del flujo de red.
- `stime`, `ltime`: Tiempo de inicio y fin del flujo.

- **Características del Contador (Counter Features)**

- `sbytes`, `dbytes`: Bytes enviados y recibidos.
- `sttl`, `dttl`: TTL (Time to Live) de paquetes enviados y recibidos.
- `sload`, `dload`: Carga enviada y recibida.

- **Características de Estadísticas (Statistical Features)**

- `spkts`, `dpkts`: Número de paquetes enviados y recibidos.
- `smean`, `dmean`: Promedio de bytes por paquete enviado y recibido.
- `sjit`, `djit`: Jitter (variación en el retardo de paquetes) enviado y recibido.

- **Características de Ataques (Attack Features)**

- `ct_flw_http_mthd`: Número de métodos HTTP.
- `ct_ftp_cmd`: Número de comandos FTP.
- `is_sm_ips_ports`: Indicador de si el mismo puerto de origen y destino es utilizado en múltiples conexiones.

Tipos de Ataques El dataset UNSW-NB15 incluye nueve categorías de ataques, que son representativas de una variedad de técnicas y tácticas utilizadas por atacantes en el mundo real. Estas categorías son:

1. **Fuerza Bruta (Fuzzers)** : Ataques que envían datos aleatorios, inválidos o inesperados a una aplicación para encontrar vulnerabilidades.
2. **Exploits** : Ataques que explotan vulnerabilidades conocidas en el software.
3. **DoS (Denial of Service)** : Ataques que intentan hacer que un servicio no esté disponible para sus usuarios.
4. **Reconocimiento (Reconnaissance)** : Ataques que buscan información sobre la red objetivo.
5. **Shellcode** : Ataques que inyectan código malicioso en la memoria de un sistema para tomar control.
6. **Backdoors** : Métodos para acceder a un sistema evitando las medidas de seguridad convencionales.
7. **Analysis** : Ataques que incluyen análisis y recolección de información.
8. **Worms** : Programas maliciosos que se replican y propagan por la red.
9. **Normal** : Tráfico legítimo y no malicioso.

Estadísticas del Dataset El dataset UNSW-NB15 contiene un total de 2,540,044 instancias, de las cuales aproximadamente 2,218,764 son tráfico normal y el resto son ataques. Para este proyecto se van a trabajar con datos limpios ya procesados que ofrece la misma página de UNSW-NB15, separados en training y testing; sin embargo, la data se muestra de forma desbalanceada. Para solucionar el problema de la data desbalanceada se desarrollará a lo largo del capítulo 3.

Capítulo 3

3. DESARROLLO DEL TRABAJO DE INVESTIGACIÓN

3.1. Tratamiento de Datos

Primero debemos saber cuántos son los datos de entrenamiento que se están realizando por lo que en primera instancia se tiene la siguiente relación (Figura 7).

```
Shape of dataframe: (257673, 43)
```

| Number of samples per attack | |
|------------------------------|-------|
| attack_cat | |
| Normal | 93000 |
| Generic | 58871 |
| Exploits | 44525 |
| Fuzzers | 24246 |
| DoS | 16353 |
| Reconnaissance | 13987 |
| Analysis | 2677 |
| Backdoor | 2329 |
| Shellcode | 1511 |
| Worms | 174 |

Figura 7: *Conteo de datos*

En la figura 7 se tiene el conteo de las instancias a transformar. Para una mejor visualización global, se realiza el siguiente diagrama (figura 8) que describe la proporción de datos :

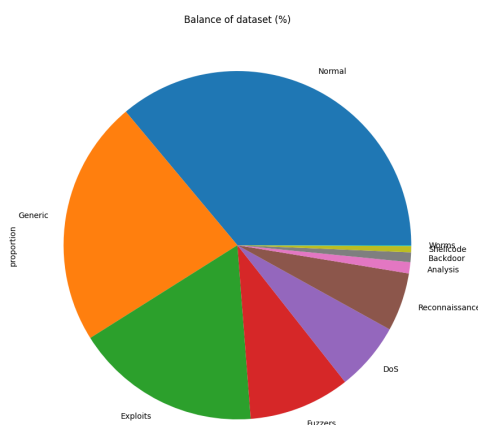


Figura 8: *Diagrama de datos*

3.2. Análisis de desequilibrio en los datos

Tanto para la etiqueta general (normal o ataque) como para los tipos específicos de ataques en ambas datasets, se tiene la siguiente distribución:

3.2.1. Distribución de la etiqueta `label` (Normal vs. Ataque)

- **Conjunto de prueba:**

- Normal (0): 37,000 instancias
- Ataque (1): 45,332 instancias

- **Conjunto de entrenamiento:**

- Normal (0): 56,000 instancias
- Ataque (1): 119,341 instancias

Distribución de los tipos de ataques (`attack_cat`)

- **Conjunto de prueba:**

- Analysis: 677 instancias
- Backdoor: 583 instancias
- DoS: 4,089 instancias
- Exploits: 11,132 instancias
- Fuzzers: 6,062 instancias
- Generic: 18,871 instancias
- Reconnaissance: 3,496 instancias
- Shellcode: 378 instancias
- Worms: 44 instancias

- **Conjunto de entrenamiento:**

- Analysis: 2,000 instancias
- Backdoor: 1,746 instancias
- DoS: 12,264 instancias
- Exploits: 33,393 instancias
- Fuzzers: 18,184 instancias
- Generic: 40,000 instancias
- Reconnaissance: 10,491 instancias
- Shellcode: 1,133 instancias
- Worms: 130 instancias

3.2.2. Diferencias en las cantidades

La clase **Normal** tiene un gran número de instancias, especialmente en el conjunto de entrenamiento, pero está equilibrada en relación a la clase de ataques.

Entre los tipos de ataque, los desequilibrios más notables se encuentran en las categorías **Generic** y **Exploits**, que tienen la mayor cantidad de instancias, mientras que **Worms** y **Shellcode** tienen relativamente pocas instancias en ambos conjuntos.

Esto puede influir en el rendimiento del modelo, ya que tendrá menos ejemplos para aprender de los tipos menos frecuentes.

Proceso realizado en el código:

Para simplificar el modelo y enfocarse en las características más relevantes, se eliminan las columnas no necesarias, como 'id' y 'label'. Esto ayuda a reducir la dimensionalidad del conjunto de datos y a eliminar información innecesaria.

Concatenación de datos

Dado que se cuenta con 2 archivos de conjuntos de datos de entrenamiento y prueba, los combinamos para manipular directamente el conjunto completo. Esto facilita la transformación y normalización de los datos de manera uniforme.

Codificación de Variables Categóricas

Las variables categóricas, como 'proto', 'state' y 'service', se transforman en valores enteros mediante la técnica de codificación de etiquetas (Label Encoding). Esta transformación es necesaria porque los algoritmos de machine learning requieren que las entradas sean numéricas.

Normalización de Datos

Se normalizan los valores numéricos para que se encuentren en un rango entre 0 y 1 utilizando la técnica de normalización Min-Max. Esto es importante para que todas las características contribuyan de manera equitativa durante el entrenamiento del modelo y se prevenga que las características con valores más grandes dominen; con ello evitamos el sesgo. Además, la normalización ayuda a que el algoritmo de aprendizaje converja más rápido y que los gradientes no exploten o desaparezcan durante el backpropagation.

Cuantización de Valores Normalizados

Después de la normalización, los valores decimales se mapean a valores enteros en un rango de 0 a 255. Esta cuantización permite representar los datos con una precisión suficiente mientras se reducen los requerimientos de almacenamiento y procesamiento.

La transformación de estos valores continuos en enteros puede parecer contraintuitiva, pero tiene varias aplicaciones prácticas como por ejemplo la reducción de la

complejidad y la reducción de Ruido y Sobrefitting.

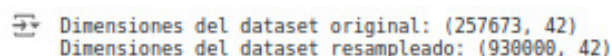
Transformación de la Columna de Etiquetas

La columna 'attack_cat', que contiene las categorías de los ataques, se convierte en variables dummy mediante one-hot encoding. Este paso es esencial para que las categorías puedan ser procesadas por el modelo de aprendizaje supervisado.

Normalización de los datos y SMOTE

Se normalizan los datos de entrada para que estén en un rango entre -0.5 y 0.5. Este paso prepara los datos para que sean compatibles con la red neuronal.

Después de la normalización, se aplica la técnica de sobremuestreo sintético (SMOTE) para generar datos adicionales para las clases minoritarias. Esto asegura que todas las clases tengan un número equilibrado de muestras.



```

→ Dimensiones del dataset original: (257673, 42)
  Dimensiones del dataset resampleado: (938888, 42)

```

Figura 9: Balanceo del conjunto de datos

3.3. Definición de modelos utilizados

En esta sección se describen los diferentes modelos utilizados para la clasificación de las secuencias temporales, especificando sus características y arquitecturas.

Modelo LSTM

Este modelo utiliza una capa LSTM con 128 unidades, seguida de una capa de regularización (*Dropout*) con una probabilidad de 0.2 para evitar el sobreajuste. La capa de salida es una capa densa con activación *softmax*, adecuada para problemas de clasificación. Los datos son organizados en lotes de tamaño 64, y se emplea el optimizador *Adam*.

Modelo GRU

El modelo GRU (Gated Recurrent Unit), este modelo emplea una capa GRU con 128 unidades, una capa de regularización (*Dropout*) con una probabilidad de 0.2, y una capa de salida con activación *softmax*.

Modelo RNN

El modelo RNN (Red Neuronal Recurrente Simple) es una red básica que utiliza conexiones recurrentes. Este modelo consta de una capa recurrente (*SimpleRNN*) con 128 unidades, seguida de una capa *Dropout* con probabilidad de 0.2 y una capa de salida con activación *softmax*.

Modelo GRU con Atención Aditiva

Este modelo combina la arquitectura GRU con un mecanismo de atención aditiva, lo que permite asignar pesos a diferentes pasos temporales según su relevancia para la predicción. Se utiliza una capa GRU con 128 unidades y una capa de atención aditiva que calcula un vector de contexto basado en las puntuaciones de atención. La salida pasa por una capa *Dropout* y una capa densa con activación *softmax*.

Modelo GRU con Atención General

El modelo GRU con atención general implementa un mecanismo de atención basado en una transformación lineal del estado oculto y las salidas del codificador. El vector de contexto generado por esta atención se combina con las salidas del codificador para realizar la predicción. La arquitectura incluye una capa GRU, una capa de atención general, y capas de regularización y salida.

Modelo GRU con Atención por Producto Punto

Este modelo incorpora un mecanismo de atención que calcula las puntuaciones mediante el producto escalar entre el estado oculto y las salidas del codificador. La atención asigna pesos para generar un vector de contexto que se utiliza para la predicción. Se emplea una capa GRU, una capa de atención por producto punto, y capas de regularización y salida.

Modelo GRU con Atención por Producto Punto Escalado

Este modelo extiende el mecanismo de atención por producto punto, escalando las puntuaciones de atención para estabilizar el entrenamiento en secuencias largas. Incluye una capa GRU para la codificación, una capa de atención escalada, y capas de regularización y salida. El vector de contexto generado se utiliza para realizar las predicciones finales.

3.4. Datos resultantes del entrenamiento

Proceso de entrenamiento

El modelo se entrena utilizando el conjunto de datos de entrenamiento (x_{train} , y_{train}) y se valida con el conjunto de datos de prueba (x_{test} , y_{test}). Se emplea la técnica de Early Stopping para detener el entrenamiento si la pérdida de validación no mejora después de cierto número de épocas, lo que ayuda a prevenir el sobreajuste y a optimizar el tiempo de entrenamiento.

Rendimiento general máximo

| Modelo | Máximo valor de exactitud | Época |
|--|---------------------------|-------|
| LSTM | 0.7351 | 59 |
| GRU | 0.7468 | 67 |
| GRU + Atención Aditiva | 0.7472 | 32 |
| GRU + Atención Producto punto | 0.7353 | 31 |
| GRU + Atención General | 0.7592 | 60 |
| GRU + Atención Producto punto escalado | 0.7093 | 15 |

Tabla 1: Valores máximos de exactitud alcanzados por cada modelo durante el entrenamiento.

Resultados de entrenamiento por modelo

| Modelo | Exactitud de entrenamiento | Exactitud de validación | Época Final |
|----------------------------|----------------------------|-------------------------|-------------|
| LSTM | 0.7085 | 0.7284 | 72 |
| GRU | 0.7465 | 0.7449 | 68 |
| GRU + Atención Aditiva | 0.7033 | 0.7224 | 45 |
| GRU + Atención Dot Product | 0.7132 | 0.7200 | 41 |
| GRU + Atención General | 0.7607 | 0.7582 | 70 |
| GRU Dot Product Escalado | 0.6926 | 0.6999 | 25 |

Tabla 2: Resultados de entrenamiento: exactitud de entrenamiento y validación, junto con la época final alcanzada para cada modelo.

Exactitud de modelos entrenados

Cada figura ilustra la evolución de la exactitud a través de las épocas de iteración para cada modelo.

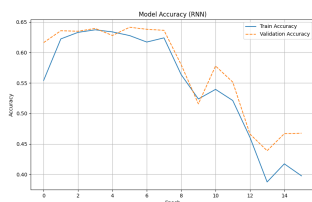


Figura 10: *Exactitud RNN*

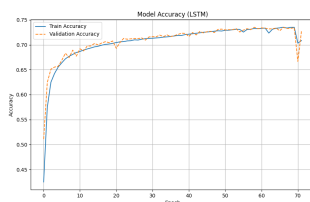


Figura 11: *Exactitud LSTM*

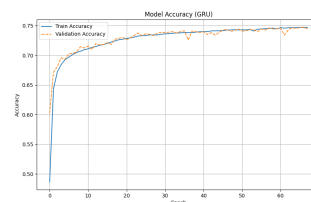


Figura 12: *Exactitud GRU*

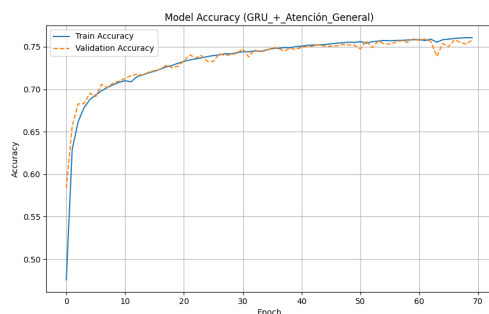


Figura 13: *Exactitud GRU + Atención general*

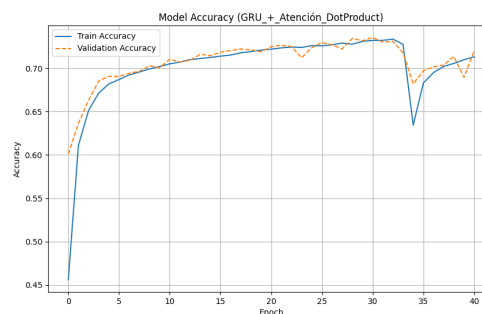


Figura 15: *Exactitud GRU + Atención producto punto*

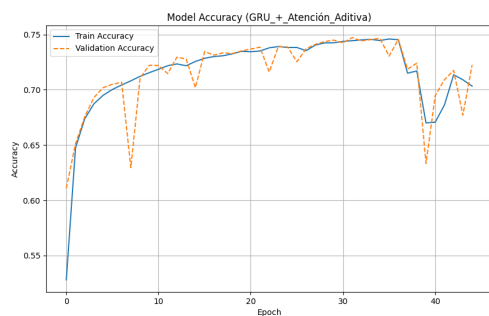


Figura 14: *Exactitud GRU + Atención aditiva*

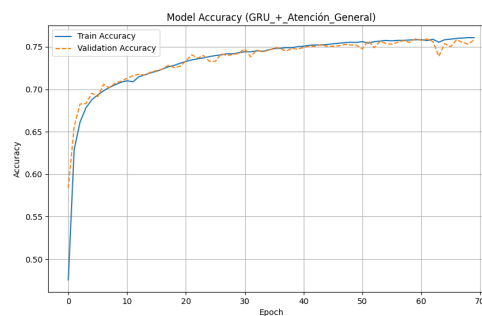


Figura 16: *Exactitud GRU + Producto punto escalado*

Pérdida de modelos entrenados

En esta sección se presentan las curvas de pérdida obtenidas durante el entrenamiento de diferentes arquitecturas de modelos. Cada figura ilustra cómo evoluciona la pérdida del modelo (loss) en función de las épocas de entrenamiento, tanto para el conjunto de entrenamiento como de validación.

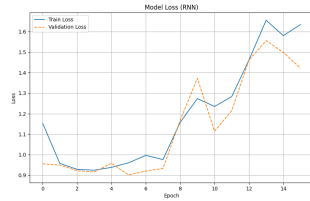


Figura 17: *Loss RNN*

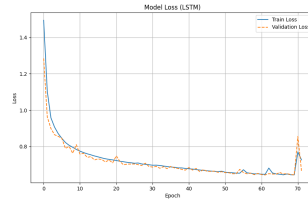


Figura 18: *Loss LSTM*

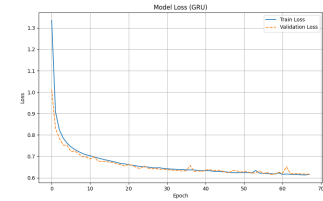


Figura 19: *Loss GRU*

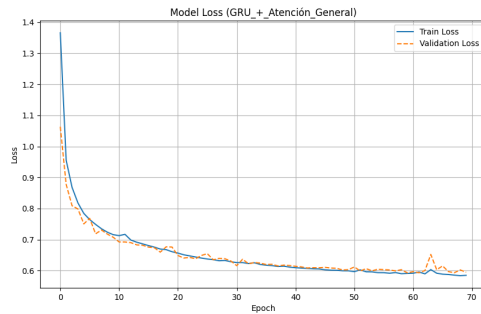


Figura 20: *Loss GRU + Atención general*

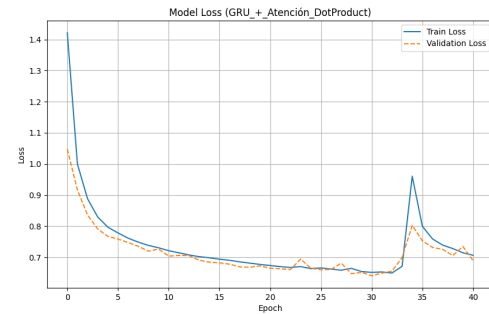


Figura 22: *Loss GRU + Atención producto punto*

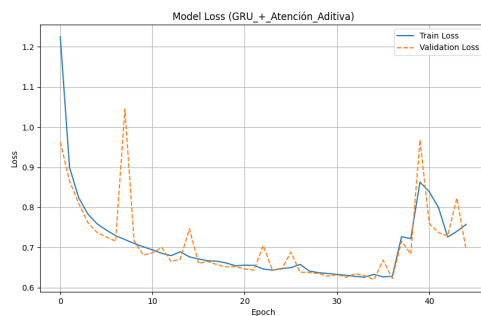


Figura 21: *Loss GRU + Atención aditiva*

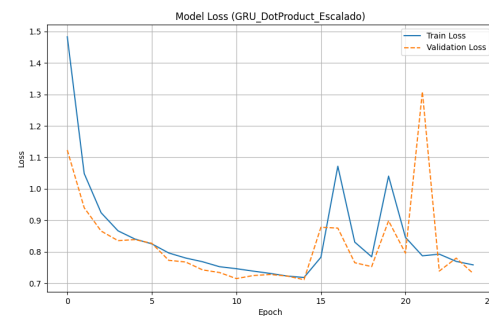


Figura 23: *Loss GRU + Producto punto escalado*

Matriz de confusión multiclase

Estas matrices permiten analizar el rendimiento de los modelos al clasificar correctamente las distintas categorías en el conjunto de datos, así como identificar los errores más frecuentes en la predicción. Cada figura muestra la distribución de las predicciones en relación con las etiquetas verdaderas, destacando las clases donde el modelo tiene mayor precisión y aquellas donde persisten confusiones significativas.



Figura 24: Matriz de confusión multiclase RNN

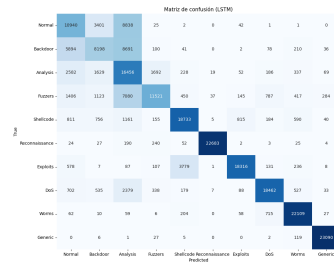


Figura 25: Matriz de confusión multiclase LSTM

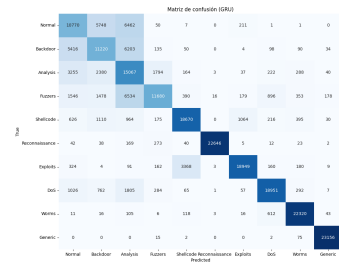


Figura 26: Matriz de confusión multiclase GRU

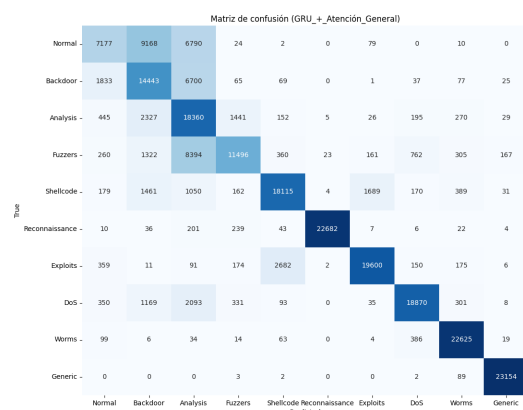


Figura 27: Matriz de confusión multiclase GRU + Atención general

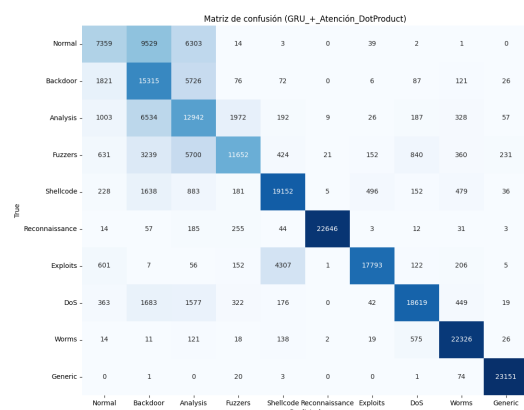


Figura 29: Matriz de confusión multiclase GRU + Atención producto punto

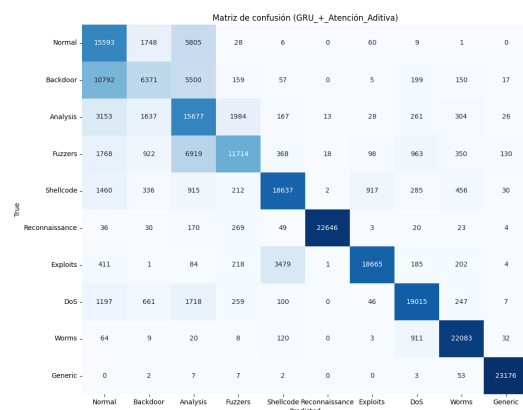


Figura 28: Matriz de confusión multiclase GRU + Atención aditiva

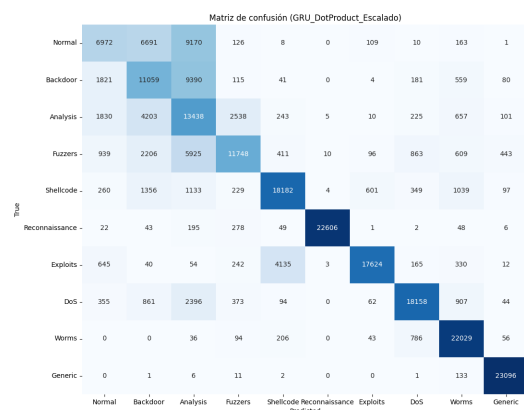


Figura 30: Matriz de confusión multiclase GRU + Atención producto punto escalado

Matriz de confusión binaria

Se presenta la capacidad de cada modelo para distinguir entre dos categorías principales: Normal y Ataque. El análisis binario simplifica la tarea de clasificación, enfocándose en la detección de comportamientos anómalos o maliciosos dentro de los datos. Las figuras muestran el número de predicciones correctas e incorrectas para cada modelo, muestra la proporción de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.

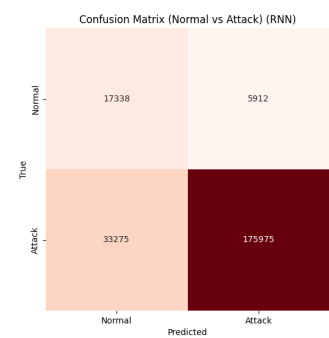


Figura 31: Matriz de confusión binaria RNN

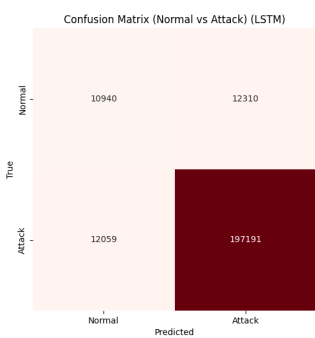


Figura 32: Matriz de confusión binaria LSTM

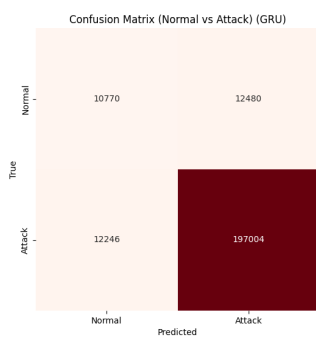


Figura 33: Matriz de confusión binaria GRU

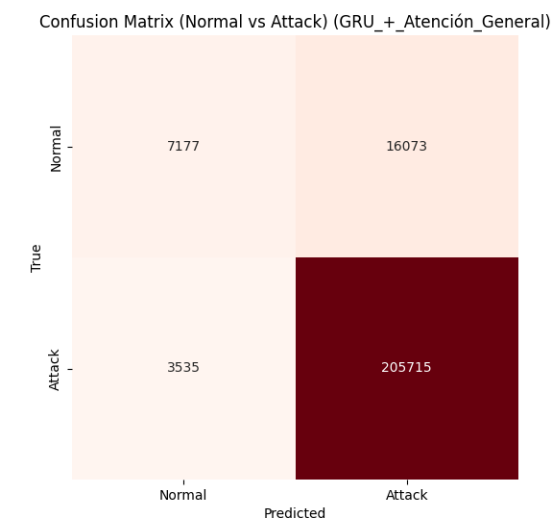


Figura 34: Matriz de confusión binaria GRU + Atención general

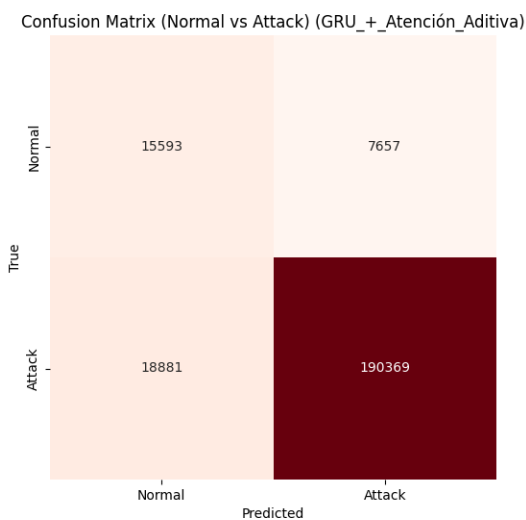


Figura 35: Matriz de confusión binaria GRU + Atención aditiva

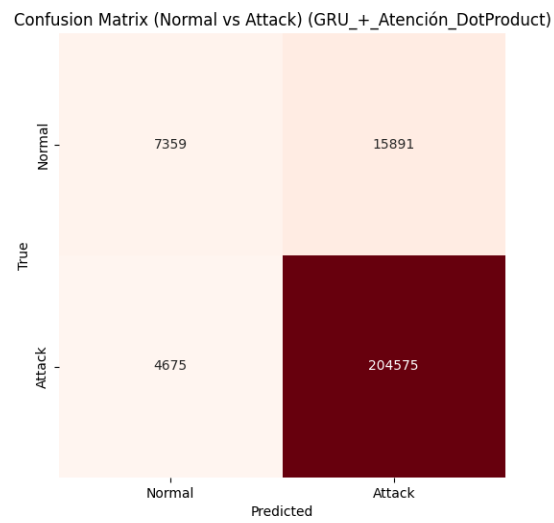


Figura 36: Matriz de confusión binaria GRU + Atención producto punto

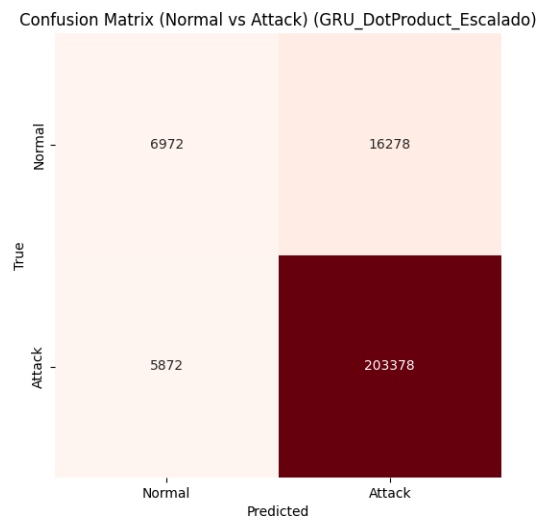


Figura 37: Matriz de confusión binaria GRU + Atención producto punto escalado

Capítulo 4

4. ANÁLISIS, DISCUSIONES DE RESULTADOS Y CONCLUSIONES

4.1. Análisis y discusiones de resultados

El desempeño de los modelos entrenados se ha evaluado en función de su precisión máxima alcanzada, el comportamiento de las métricas de entrenamiento y validación, y el análisis de las matrices de confusión.

Comparación entre RNN, LSTM y GRU:

Entre los modelos base (RNN, LSTM y GRU), se observó que el modelo GRU alcanzó la mayor precisión tanto en entrenamiento (0.7465) como en validación (0.7449), superando a LSTM y RNN. Esto se alinea con las características de las GRU, que tienen una arquitectura más eficiente en el manejo de dependencias a largo plazo en comparación con las RNN tradicionales.

Aunque el modelo LSTM mostró un buen desempeño con una precisión de validación cercana a 0.7351, requirió más épocas (59) para converger en comparación con GRU.

Impacto de los mecanismos de atención en GRU:

Los modelos que combinan GRU con mecanismos de atención mostraron un incremento significativo en la precisión en comparación con la GRU básica. En particular, GRU + Atención General alcanzó la mayor precisión de validación (0.7592) en la época 60.

La atención aditiva y la atención por producto punto también mostraron mejoras, con precisiones de validación de 0.7472 y 0.7353, respectivamente. Esto demuestra que los mecanismos de atención contribuyen a una mejor identificación de patrones relevantes en los datos, permitiendo al modelo priorizar información importante.

El modelo GRU + Atención producto punto escalado, aunque mostró un desempeño inferior (0.7093), converge más rápido (15 épocas), lo cual puede ser útil en escenarios donde el tiempo de entrenamiento es importante en cuestión de reapidéz y precisión.

Matriz de confusión multiclase y binaria:

Las matrices de confusión revelaron que los modelos con atención logran reducir los errores en clases críticas, destacando una mejor capacidad para distinguir entre categorías complejas.

4.2. Conclusiones

- Los modelos GRU demostraron un mejor rendimiento general en comparación con LSTM y RNN, validando su eficiencia para tareas de clasificación.
- La integración de mecanismos de atención en las GRU mejora significativamente la precisión de los modelos al permitir un enfoque más detallado en características relevantes. El modelo GRU + Atención General sobresale como el más eficiente, logrando la mayor precisión de validación (0.7592).
- Aunque los modelos GRU + Atención producto punto escalado tienen un menor rendimiento en precisión, ofrecen una ventaja en términos de velocidad de convergencia, siendo adecuados para aplicaciones donde el tiempo de entrenamiento es importante.

REFERENCIAS

- [1] CHECKPOINT, “cyber security report 2023,” 2023. [Online]. Available: <https://pages.checkpoint.com/cyber-security-report-2023.html#:~:text=The%202023%20Cyber%20Security%20Report%20gives%20a%20detailed%20synopsis%20of,attack%20%E2%80%93%20prevention%20is%20at%20reach!>
- [2] M. Zurita, “¿qué tanto han avanzado los ciberataques y qué estrategias pueden adoptar las empresas que operan en Perú para enfrentarlos?” Feb 2024. [Online]. Available: <https://forbes.pe/especiales/2024-02-15/que-tanto-han-avanzado-los-ciberataques-y-que-estrategias-pueden-adoptar-las-empresas-que-operan-en-peru>
- [3] IBM, “¿qué es la detección y respuesta de red (ndr)?” Dec 2023. [Online]. Available: <https://www.ibm.com/es-es/topics/ndr>
- [4] Fortinet, “what is intrusion detection systems (ids)? how does it work?” 2022. [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/intrusion-detection-system>
- [5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, Jul 2019. [Online]. Available: <https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7#citeas>
- [6] GeeksforGeeks, “Introduction to deep learning,” Jun 2018. [Online]. Available: <https://www.geeksforgeeks.org/introduction-deep-learning/>
- [7] M. T. J. Samaya Madhavan, “Deep learning architectures,” Jan 2021. [Online]. Available: <https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/>
- [8] F. Tanco, *Redes Neuronales Artificiales*, Feb 2021. [Online]. Available: <https://www.frba.utn.edu.ar/wp-content/uploads/2021/02/RNA.pdf>
- [9] A. W. Services, “¿qué es una red neuronal?” 2023. [Online]. Available: <https://aws.amazon.com/es/what-is/neural-network/>
- [10] M. Benito, “¿qué es el deep learning y cuáles son sus aplicaciones?” Sep 2020. [Online]. Available: <https://fp.uoc.fje.edu/blog/que-es-deep-learning-y-cuales-son-sus-aplicaciones/>
- [11] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark,” *Neurocomputing*, vol. 503, p. 92–108, Sep 2022.

- [12] G. Brauwiers and F. Frasincar, “A general survey on attention mechanisms in deep learning,” *IEEE Transactions on Knowledge and Data Engineering*, p. 1–1, 2022.
- [13] U. Tewari, “Regularization — understanding l1 and l2 regularization for deep learning,” Nov 2021. [Online]. Available: <https://medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning-a7b9e4a409bf>
- [14] M. L. A. J. S. Aston Zhang, Zachary C. Lipton, “Redes neuronales recurrentes,” 2021. [Online]. Available: https://d2l.ai/chapter_recurrent-neural-networks/rnn.html
- [15] I. D. Mienye, T. G. Swart, and G. Obaido, “Recurrent neural networks: A comprehensive review of architectures, variants, and applications,” *Information*, vol. 15, no. 9, 2024. [Online]. Available: <https://www.mdpi.com/2078-2489/15/9/517>
- [16] Jeremy, “Understanding recurrent networks (part 1 — simple rnn),” Oct 2023. [Online]. Available: <https://medium.com/@imjeremyhi/understanding-recurrent-networks-part-1-simple-rnn-lstm-cc53e7475980>
- [17] S. Saxena, “Introduction to lstm (long short term memory),” Aug 2023. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>
- [18] Ryan, “Lstms explained: A complete, technically accurate, conceptual guide with keras,” Sep 2020. [Online]. Available: <https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2>
- [19] M. L. A. J. S. Aston Zhang, Zachary C. Lipton, “Dive into deep learning,” 21 2021. [Online]. Available: https://d2l.ai/chapter_recurrent-modern/lstm.html
- [20] A. A. Awad, A. F. Ali, and T. Gaber, “An improved long short term memory network for intrusion detection,” *PloS one*, vol. 18, no. 8, p. e0284795–e0284795, Aug 2023. [Online]. Available: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0284795>
- [21] “An intuitive explanation of lstm - ottavio calzone - medium.” [Online]. Available: <https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c>
- [22] “Introduction to gated recurrent unit (gru),” Mar 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/>
- [23] IBM, “Red neuronal recurrente (rnn),” Oct 2021. [Online]. Available: <https://www.ibm.com/es-es/topics/recurrent-neural-networks>

- [24] “Gru,” May 2024. [Online]. Available: <https://miguelevangelista.gitbook.io/inteligencia-artificial/redes-neronales-recurrentes/arquitecturas/gru>
- [25] M. L. A. J. S. Aston Zhang, Zack C. Lipton, “Gated recurrent units (gru),” 2024. [Online]. Available: https://classic.d2l.ai/chapter_recurrent-modern/gru.html#fig-gru-1
- [26] D. Bahdanau, K. Cho, and Y. Bengio, *Published as a conference paper at ICLR 2015 NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE*, 2016. [Online]. Available: <https://arxiv.org/pdf/1409.0473>
- [27] M.-T. Luong, H. Pham, and C. Manning, *Effective Approaches to Attention-based Neural Machine Translation*, 2015. [Online]. Available: <https://arxiv.org/pdf/1508.04025>
- [28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Kaiser, and I. Polosukhin, *Attention Is All You Need*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fb0d053c1c4a845aa-Paper.pdf
- [29] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, no. 16, Jun 2002.
- [30] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-smote: A new over-sampling method in imbalanced data sets learning,” *LNCS*, vol. 3644, 2005. [Online]. Available: <https://sci2s.ugr.es/keel/keel-dataset/pdfs/2005-Han-LNCS.pdf>
- [31] A. Tanda, “Native bees are important and need immediate conservation measures: A review †,” 2021. [Online]. Available: <https://sciforum.net/manuscripts/10523/manuscript.pdf>
- [32] E. A. G. S. L. Haibo He, Yang Bai, “adasyn: adaptive synthetic sampling approach for imbalanced learning,” *ResearchGate*, July 2008. [Online]. Available: https://www.researchgate.net/publication/224330873_ADASYN_Adaptive_Synthetic_Sampling_Approach_for_Imbalanced_Learning
- [33] ACCS, “the unsw-nb15,” 2015. [Online]. Available: <https://research.unsw.edu.au/projects/unsw-nb15-dataset>