Universidad Michoacana de San Nicolás de Hidalgo
Facultad de Ingeniería Eléctrica

Aprendizaje por transferencia y sintonización fina para la identificación y clasificación de eventos en la calidad de energía

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Maestro en Ciencias EN Ingenieria Electrica

P R E S E N T A :

Ing. Miguel Gabriel Juárez Jiménez

DIRECTOR DE TESIS

Dr. Jaime Cerda Jacobo

CO-DIRECTOR DE TESIS

Dr. Alejandro Zamora Méndez

Morelia, Michoacán, DICIEMBRE 2024

Agradecimientos

Resumen

En este trabajo de tesis se abordá la clasificación y detección de eventos en la calidad de la energía (PQ), incluyendo 28 fenómenos en los cuales se encuentran: armónicos, flicker, notch, sag swell y varios eventos complejos derivados de estos. Se define matemáticamente cada uno de los 28 eventos a clasificar, comprendiendo tanto los simples como las eventos complejos, junto con ello se desarrolla una herramienta para crear señales sintéticas, siguiendo el modelo matemático.

Se pre-entrenan 4 modelos de redes neuronales convolucionales para la extracción de características que posteriormente se concatenan para crear un modelo mas grande y robusto al cual se le aplican técnicas de transferencia de aprendizaje y sintonizacion fina. Una vez que se obtiene el modelo al que se le aplican las técnicas de sintonizacion fina y transferencia de aprendizaje, se prueba y valida con señales sintéticas con las cuales alcanza una eficiencia del 98 %. Ya que se ha validado el modelo se prueba con dos grupos de señales, señales simuladas provenientes de 3 modelos de simulink, y con señales reales.

Una vez terminados los resultados, se detallan los beneficios obtenidos al realizar este trabajo de tesis, las ventajas y posibles desventajas de aplicar este tipo de modelos y técnicas para la identificación y clasificación de eventos (PQ), así como los posibles trabajos futuros que se pueden implementar gracias a este trabajo.

Palabras clave: Sistemas renovables de potencia, aprendizaje profundo, eventos PQ, redes neuronales convolucionales, transferencia de aprendizaje, sintonización fina

Abstract

This thesis addresses the classification and detection of events in power quality (PQ), including 28 phenomena in which: nicos, flicker, notch, sag swell and various complex events derived from these. Each of the 28 events to be classified is mathematically defined, including both simple and complex events. Along with this, a tool is developed to create synthetic signals, following the mathematical model.

4 convolutional neural network models are pre-trained for feature extraction that are subsequently concatenated to create a larger and more robust model to which transfer learning and fine tuning techniques are applied. Once the model to which the fine tuning and transfer learning techniques are applied is obtained, it is tested and validated with synthetic signals with which it reaches an efficiency of 98Since the model has been validated, it is tested with two groups of signals, simulated signals from 3 simulink models, and with real signals.

Once the results are finished, the benefits obtained by carrying out this thesis work are detailed, the advantages and possible disadvantages of applying this type of models and techniques for the identification and classification of events (PQ), as well as the possible future works that can be implemented thanks to this work.

Keywords: Renewable power systems, deep learning, PQ events, convolutional neural networks, transfer learning, fine tuning

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Planteamiento del problema

En los últimos años, las redes interconectadas de energía renovable en muchos países se están desarrollando rápidamente, lo que lleva a la instalación de microrredes de energía renovable que son paralelas a las redes eléctricas tradicionales. Sin embargo, los sistemas eléctricos de potencia (SEPs) son cada vez más grande y el uso de dispositivos basados en electrónica de potencia conducen a una serie de perturbaciones en la calidad de la energía (PQ, del inglés Power Quality) en los SEPs.

Por lo tanto, es crucial identificar los orígenes y desencadenantes de los eventos que afectan la calidad de la energía en los SEPs para así implementar medidas preventivas. Para comprender las causas y el origen de estas alteraciones, es importante contar con un monitoreo y análisis para la identificación y prevención de este tipo de eventos PQ. Por lo tanto, el monitoreo de estos eventos PQ se ha convertido en una necesidad para una rápida detección y rectificación de las perturbaciones de dichos eventos.

El problema principal que se trata en esta tesis, es la identificación mediante técnicas de aprendizaje profundo (del inglés deep learning) tal que nos permita realizar una identificación eficaz y eficiente de eventos PQ en SEPs.

1.2. Antecedentes

Como resultado de la integración masiva de la generación distribuida, la agregación de energías renovables, la proliferación de cargas no lineales y el uso de dispositivos de electrónica de potencia ampliamente desplegados en los sistemas eléctricos, ha provocado un deterioro en la calidad de la energía en los sistemas de potencia. Los eventos resultantes en este deterioro en la calidad de la energía se detallan en las normas [2, 3, 4, 5, 6] y se mencionan brevemente a continuación:

- **Sags**. Es una caída de tensión, o en otras palabras, es una breve reducción en la tensión RMS de 10 % o más por debajo de la tensión especificada (nominal) del equipo durante un período de 1/2 ciclo a 1 min, como se define en la norma IEC 61000-4-30 [7].

- **Swell**. Es lo opuesto a las caídas de tensión y se definen como un aumento momentáneo en la tensión RMS del 10 % o más, por encima de la tensión (nominal) del equipo durante un período de 1/2 ciclo a 1 min, como se define en la norma IEC 61000-4-30 [7].

- **Interrupción**. Es un evento durante el cual, el voltaje en el punto de conexión del usuario cae a cero y no retorna a sus valores normales automáticamente. De acuerdo con la IEC [7], el tiempo mínimo de una larga interrupción es de 3 minutos. Si el tiempo es menor a 3 min, se denomina de corta interrupción.

- **Impulso o transitorio**. Son eventos repentinos de cresta alta que elevan la tensión y/o los niveles de corriente en dirección positiva o negativa. Los transitorios impulsivos pueden ser eventos muy rápidos (5 ns de tiempo de ascenso desde estado estable hasta el valor máximo del impulso) de una duración breve (menor a 1 ciclo).

- **Transitorio oscilatorio**. Es un cambio repentino, a distinta frecuencia de la frecuencia fundamental, de la condición estacionaria de la tensión, la corriente o ambos

y que tiene valores de polaridad positivos y negativos.

- **Armónicas e inter-armónicas**. Significa que existen señales superpuestas a la señal fundamental (60 Hz para los casos aquí propuestos), cuyas frecuencias, para el caso de las armónicas, corresponden a múltiplos enteros de la frecuencia fundamental, en el caso de las inter-armónicas corresponde a múltiplos no enteros de la frecuencia fundamental. Estas frecuencias provocan la variación de la forma de onda de voltaje y corriente de la forma senoidal ideal, creando perturbaciones en la calidad de energía, tales como la carga de potencia reactiva y baja eficiencia en el sistema.

- **Flicker de electricidad**. Es una variación perceptible por el ojo humano en la luminosidad e intensidad de la luz debido a fluctuaciones o variaciones de tensión en la red eléctrica. Se trata de una especie de parpadeo o reducción de la luz de manera intermitente.

- **Notch o notching**. Es una perturbación periódica de voltaje causada por el funcionamiento normal de los dispositivos electrónicos de potencia cuando la corriente se conmuta de una fase a otra.

- **Eventos complejos**. Son una combinación de los eventos mencionados anteriormente, por ejemplo, una señal con contaminación armónica y un sag o un swell con un transitorio oscilatorio. Cabe resaltar, que estas combinaciones pueden llevar más de 2 eventos.

Cada evento tiene sus propios efectos en la calidad y el deterioro de la energía eléctrica, aquí se en lista de manera general algunos efectos resultantes de una mala calidad de la energía [8]:

- Fallo del banco de capacitores por avería dieléctrica o sobrecarga de potencia reactiva [9, 10].

- Interferencia con el control de ondulación y los sistemas portadores de la línea eléctrica, lo que causa un mal funcionamiento en sistemas que realizan conmutación remota, control de carga y medición [11].

- Pérdidas excesivas y calentamiento de máquinas síncronas y de inducción [12].

- Sobrevoltajes y sobrecorrientes excesivas en el sistema desde resonancia hasta voltajes armónicos o corrientes en la red [13].

- Ruptura dieléctrica de cables aislados como resultado de sobrevoltajes armónicos en el sistema [14].

- Interferencia inductiva con sistemas de telecomunicaciones [15].

- Errores en inducción de medidores kWh [16].

- Interferencia de señal y mal funcionamiento en relés, particularmente en sistemas de estado sólido y controlados por microprocesador [17].

- Interferencia con controladores de motores grandes y sistemas de excitación de centrales eléctricas [18].

- Oscilaciones mecánicas en máquinas de inducción y síncronas [19].

- Funcionamiento inestable de los circuitos de encendido basado en la detección de cruce de voltaje por cero o enclavamiento. Estos efectos dependen de la fuente de armónicas y su ubicación en el sistema de potencia, y las características de la red que promueven la propagación de armónicas [20].

- Sobrecalentamiento de los conductores neutros [21].

- Perturbaciones en los interruptores automáticos de los circuitos [22].

- Una mala calidad de la energía degradará, sin que el usuario lo perciba, los componentes electrónicos y los circuitos, reduciendo la vida efectiva de los equipos y aumentando los fallos [23].
- Disrupción del sistema eléctrico [24].
- Numerosos tipos de daños y pérdidas económicas en componentes electrónicos [21, 22, 23, 24].

Uno de los métodos más comúnmente utilizado para la clasificación e identifiación de eventos en la calidad de la energía (PQ), es la transformada de Fourier, la transformada Wavelet, una combinación entre la transformada Wavelet y la transformada de Fourier conocida como S-Transform o transformada de Stockwell, algoritmos de machine learning, entre otros.

Algunos de los trabajos principales relacionados a este trabajo de tesis son los siguientes:

- En [25] se presentan revisiones exhaustivas del procesamiento de señales y técnicas inteligentes para la clasificación automática de eventos de calidad de energía (PQ) y el efecto del ruido en la detección y clasificación de perturbaciones.
- En [26], presentan una combinación del método del campo de suma angular de Gramian (GASF) con una red neuronal convolucional (CNN). En el cual, una señal de perturbación de la calidad de la energía 1-D se transforma en un archivo de imagen 2-D utilizando GASF, se implementa CNN para la extracción de características y clasificación de imágenes. Se consideran perturbaciones sintéticas, incluidas nueve perturbaciones simples y cinco perturbaciones mixtas.
- El trabajo en [27] presenta un árbol de decisión en combinación con la matriz de amplitud de la transformada S. Las características obtenidas de la matriz de amplitud de la transformada S son diferentes, claras e inmunes al ruido. Según un árbol de

decisión basado en reglas, en este trabajo se identifican bien 7 tipos de perturbaciones eléctricas simples y 16 tipos de perturbaciones eléctricas complejas.

- De forma similar, [28] presenta un análisis multidimensional, estadísticas de orden superior con un clasificador basado en un neuro árbol un árbol de decisión que utiliza al perceptrón en cada nodo para la extracción de características. Esta implementación permite la ejecución en tiempo real y su aplicación para monitorear redes inteligentes. Es capaz de detectar desviaciones en la forma de onda de tensión medida respecto a su valor nominal y puede clasificar 20 clases de perturbaciones simples y múltiples con una eficiencia global superior al 97 %.

- De igual forma, existen trabajos como en [29], que utilizan un enfoque de circuito cerrado completo para detectar y clasificar perturbaciones en la calidad de la energía basado en una red neuronal convolucional profunda, clasificando hasta 16 perturbaciones simples y complejas.

- En [30] presentan un codificador automático apilado, como marco de aprendizaje profundo, se emplea para extraer características de alto nivel de los eventos PQ para su clasificación. En este trabajo se aplican variaciones de señales y un algoritmo de optimización de enjambre de partículas para ayudar en la clasificación de los PQ.

- En [31] utilizan redes neuronales convolucionales que utilizan escalogramas como imágenes de entrada para llevar a cabo la tarea de clasificación. Los escalogramas se generan mediante extracción de características mediante la transformada wavelet.

- En [32], presentan una red neuronal probabilística (PNN) basado en una extracción de características con la transformada S para el reconocimiento de perturbaciones en la calidad de la energía. Los resultados de la simulación revelan que la combinación de la transformada S y PNN puede detectar y clasificar eficazmente 11 diferentes eventos PQ.

De lo anterior, la contribución de esta tesis radica en la eficiencia y simpleza del método propuesto, ya que sin realizar ningún pre-procesamiento, el método propuesto es capaz de detectar y clasificar 28 eventos, tanto complejos como simples de calidad de la energía, siendo el más complejo una combinación de hasta 4 eventos.

1.3. Justificación

Debido al uso cada vez mayor de sistemas de energía renovable, tales como la generación de energía eólica y solar, así como la utilización de la electrónica de potencia en éstos sistemas de generación distribuida; además de otros dispositivos eléctricos tales como los hornos de arco eléctrico utilizados en la industria, detectar eventos en la calidad de la energía es crucial por varios motivos fundamentales tales como [33, 34]:

- Mantenimiento de Equipos [35]: La detección temprana de eventos como fluctuaciones de voltaje, sobrecargas o armónicos ayuda a prevenir daños en equipos eléctricos sensibles. Identificar estos problemas a tiempo permite un mantenimiento preventivo, prolongando la vida útil de los dispositivos y reduciendo costos de reparación o reemplazo.

- Continuidad del Servicio [36]: Los eventos en la calidad de la energía pueden causar interrupciones en el suministro eléctrico. Detectarlos permite implementar medidas correctivas para evitar apagones o fallos en la red eléctrica, asegurando la continuidad del servicio tanto en entornos industriales como residenciales.

- Eficiencia Energética: La identificación de eventos como pérdidas de energía o sobrecargas no solo ayuda a mantener la estabilidad de la red, sino que también contribuye a una mejor gestión de la energía. Esto implica un uso más eficiente de los recursos, lo que a su vez reduce costos operativos y disminuye el impacto ambiental.

- Calidad de Producción: En entornos industriales, la calidad de la energía puede afec-

tar la maquinaria y la producción. Detectar y corregir problemas de calidad eléctrica garantiza un entorno más estable para la fabricación, mejorando la calidad de los productos y reduciendo la probabilidad de fallas en la producción.

- Seguridad: Eventos como picos de voltaje o armónicos pueden representar riesgos para la seguridad de las personas y los equipos conectados. Detectar estos eventos es fundamental para prevenir accidentes, proteger a las personas y garantizar entornos de trabajo seguros.

En resumen, la detección de eventos en la calidad de la energía es esencial para mantener la eficiencia, seguridad y continuidad en la distribución y el uso de la electricidad, tanto en entornos residenciales como industriales.

1.4. Objetivos

1.4.1. Objetivos generales

La contribución principal de esta investigación radica en la propuesta de técnicas de aprendizaje profundo para la identificación y correcta clasificación de los eventos de calidad de la energía. Para esto, se utiliza una red neuronal convolucional en una dimensión, cuya ventaja es de que no se requiere realizar un preprocesamiento a las señales de entrada de la propuesta. Además, se aplican técnicas de aprendizaje por transferencia y sintonización fina para lograr una mejor identificación y clasificación de los eventos. Así, la combinación de estas tres técnicas se prueban exahustivamente para la identificación de eventos de calidad de energía, obteniéndose buenos resultados. Además, una de las ventajas principales de la metodología propuesta, es que se puede aplicar directamente a datos , requiriendo recursos computacionales moderados, por lo que sería posible una implementación en tiempo real.

1.4.2. Objetivos particulares

1. Clasificación de eventos de calidad de la energia: Caracterizar 28 tipos de eventos. En una primera etapa, se consideran eventos simples tales como Sags, Swell, Interrupciones, Armónicos, Flicker, Notch, Transitorios Oscilatorios, Impulso/Spike. Posteriormente, modelar eventos complejos mediante la combinación de 2 a 4 eventos simples.

2. Desarrollo de herramientas y generación de datos: construir una herramienta en Python para generar señales sintéticas basadas en modelos matemáticos. A partir de las señales generadas entrenar y validar modelos basados en redes neuronales. Adicionalmente, utilizar conjuntos de datos obtenidos a partir de mediciones en sistemas reales para validar los modelos ajustados a partir de las señales sintéticas.

3. Entrenamiento y diseño del modelo: Implementar un modelo mediante basado en redes neuronales convolucionales, el cual integre el uso de técnicas de IA tales: como transferencia de aprendizaje, regularización, ajuste fino. Adicionalmente, se implemetará una combinación modelos ajustados para en la clasificación de las diferentes combinaciones de eventos propuestas.

4. Validación y comparación: Validar los modelos propuestos, esto mediante el uso de técnicas como la validación cruzada sobre diversos conjuntos de datos, tanto reales, como sintéticos con y sin ruido gaussiano; así como una comparación de los resultados obtenidos con estudios del estado del arte.

1.5. Descripción de capítulos

En este Capítulo se presentan los antecedentes, justificación y planteamiento de la problemática, lo que permite desarrollar este trabajo de investigación, así como una breve descripción de cada Capítulo. El Capítulo 2 describe la formulación matemática del

método propuesto y las técnicas utilizadas para mejorar su eficiencia, también se presenta la formulación matemática de las señales sintéticas para su correcta simulación. El Capítulo 3 presenta la aplicación de los algoritmos propuestos y el resultado que estos arrojan para la identificación y clasificación de eventos PQ en SEPs. Finalmente, en el Capítulo 4 se presentan las conclusiones y posibles trabajos futuros a realizar.

Capítulo 2

Fundamentos matemáticos

2.1. Aprendizaje Profundo

El aprendizaje profundo es un subconjunto del aprendizaje automático (Machine Learning, ML), que a su vez es un subconjunto de la inteligencia artificial (Artificial Intelligence, IA). En el aprendizaje profundo los algoritmos se inspiran en el cerebro humano y se conocen como redes neuronales, tienen como objetivo emular el proceso que se daría en un cerebro humano para llegar a un resultado.

2.1.1. Perceptrón

El perceptrón se puede ver como una red neuronal con una sola capa y una sola neurona, matemáticamente el perceptrón se puede describir como [37]:

$$a = \sum_{j=1}^{n} w_j x_j + b = \vec{w} \cdot \vec{x} + b \tag{2.1}$$

donde w_j son los pesos o los parámetros a optimizar, x_j es la $j_e sima$ característica de entrenamiento, b es el sesgo o bias y n nos indica el numero de características de los datos utilizados. La Figura 2.1 muestra de manera gráfica el perceptrón, la unica diferencia con la Ecuación 2.1 es que se agrega la función de activación $f(a)$ que se menciona mas delante.

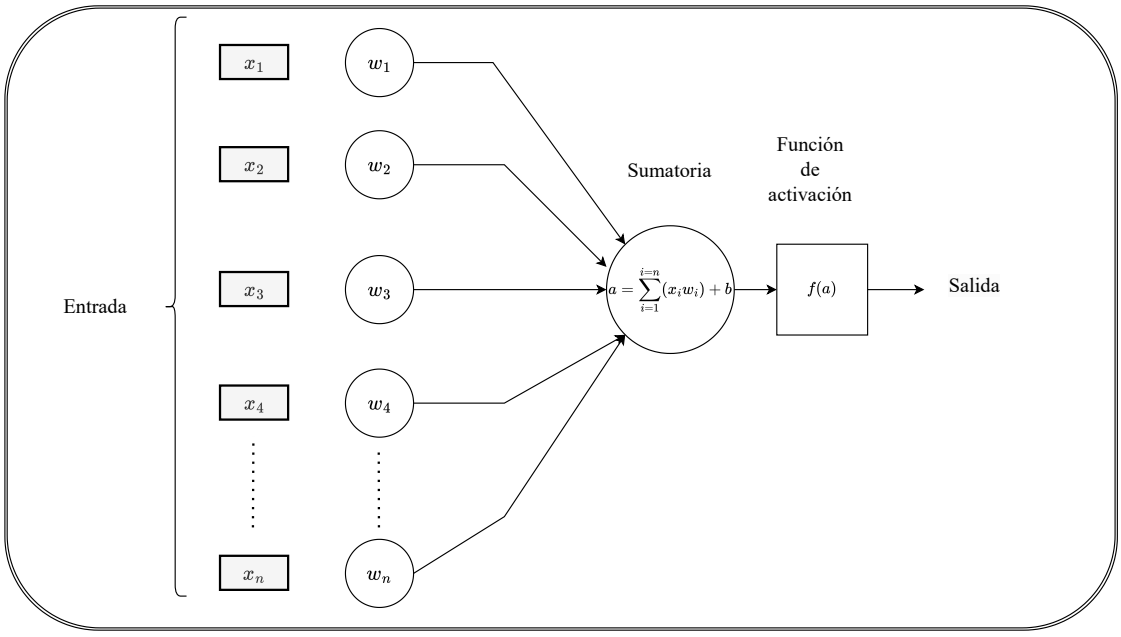


Figura 2.1: Perceptrón.

Como se menciona anteriormente, (2.1) es equivalente a una red neuronal con una sola capa y una sola neurona, para agregar neuronas a esta única capa y en lugar de un perceptrón tener un MLP (Multi-layer peceptron), basta con agregar más vectores de pesos $\vec{w}$ al producto en (2.1), por lo que ahora la MLP se representa como [38]:

$$a_j = \sum_{i=1}^{n} w_{(j,i)} x_i + b_j \tag{2.2}$$

que en forma matricial queda como:

$$\vec{a} = \vec{x} \cdot \mathbf{w} + \vec{b} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix} \begin{bmatrix} w_1^{(1)} & w_1^{(2)} & \cdots & w_1^{(N)} \\ w_2^{(1)} & w_2^{(2)} & \cdots & w_2^{(N)} \\ w_3^{(1)} & w_3^{(2)} & \cdots & w_3^{(N)} \\ \vdots & \vdots & \vdots & \ddots \\ w_n^{(1)} & w_n^{(2)} & \cdots & w_n^{(N)} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & b_3 & \vdots & b_N \end{bmatrix} \tag{2.3}$$

donde el sub-índice n nos indica el número de características del conjunto de datos, el súper-índice (N) nos indica el numero de neuronas o perceptrónes que vamos a utilizar y el vector $\vec{a}$ nos indica las posibles clases en las que se puede clasificar o diferenciar con el MLP, note que entre más neuronas, en teoría, se puede diferenciar o "aprender"más clases o características en los datos. En (2.3) se da la intuición de como agregar neuronas, basta con agregar columnas a la matriz de pesos, es decir, si la matriz W tiene un tamaño $N \times n$ y queremos aumentarle una neurona, basta con hacerla de tamaño $(N+1) \times n$, para agregar una capa completa, basta con multiplicar el vector resultante $\vec{a}$ con una nueva matriz o vector de pesos, de esta manera se tienen dos capas, o K capas, matemáticamente se puede representar como [38]:

$$\vec{a} = [[\vec{x} \cdot W_1 + \vec{b}] \cdot W_2 + \vec{b}] ... \cdot W_k + \vec{b} \tag{2.4}$$

Agregando la función de activación, que se menciona en la Sección 2.1.2 a (2.4), se muestra básicamente el algoritmo de propagación hacia adelante (al realizar la multiplicación matricial) o la forma en la que se predicen las clases una vez que la red se entrena y se optimiza para ser utilizada. Así, re-escribiendo (2.4) de la siguiente manera:

$$\vec{a} = \vec{x} \cdot \underbrace{W_1 W_2 ... \cdot W_k}_{W} + \underbrace{\vec{b} \cdot W_2 ... \cdot W_k + \vec{b} \cdot W_k + \vec{b}}_{b} \tag{2.5}$$

se puede observar, que si se agregan varias capas, el resultado final es una ecuación lineal y el resultado seria equivalente a tener una sola matriz, ya que la aproximación sería equivalente a tener un solo perceptrón, lo que lleva a un gasto computacional muy alto y poco eficiente en el modelo, es por esto, que en la Sección 2.1.2 se muestra la importancia de la función de activación, ya que la correcta utilización de eśta, ayuda a mitigar las posibles problemáticas que se presentan.

2.1.2. Función de activación

La función de activación puede tener varios propósitos. En esta tesis únicamente se mencionan dos propósitos: (*i*) para una aproximación adecuada hacia los datos y (*ii*)

para evitar cálculos innecesarios.

Ejemplos de las problemáticas

Uno de los propósitos más importantes de la función de activación es la aproximación hacia los datos. La demostración más grafica que se puede realizar es un caso de regresión, ya que típicamente los datos no son lineales, pero aplica de la misma manera para clasificación o alguna de las otras áreas de Machine Learning.

Por ejemplo, la Figura 2.2 muestra un grupo de datos con un comportamiento lineal, en este caso, al aplicar una red neuronal sin función de activación, la red neuronal puede predecir el comportamiento de estos datos de manera adecuada y en este caso no es necesaria una función de activación.

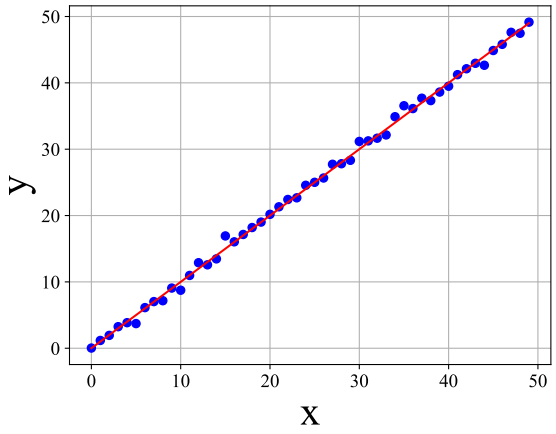


Figura 2.2: Datos lineales y su aproximación lineal.

Ahora, la pregunta es ¿qué pasa si el comportamiento del conjunto de datos no es lineal?, esto se muestra en la Figura 2.3. Como se puede observar, la aproximación lineal no es una aproximación adecuada para este conjunto de datos, el error que existe es muy grande y al tratar de predecir el comportamiento de un nuevo dato es muy posible que el modelo se equivoque.

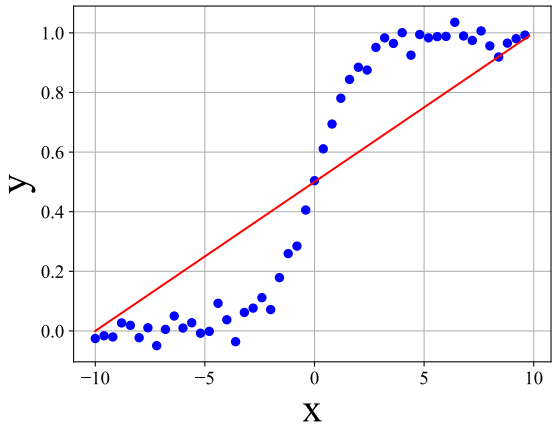


Figura 2.3: Datos NO lineales y una aproximación lineal.

De lo anterior, se observa que es necesaria una función de activación o una función que pueda aproximar los datos de manera adecuada, en este caso particular, el conjunto de datos es similar a la función sigmoide. Así, al aplicar esta función a la salida de la red neuronal, la aproximación hacia los datos es aun mayor, por lo que utilizar esta función minimiza el error y aumenta la posibilidad de predecir un nuevo dato de manera más adecuada. La Figura 2.4 muestra el conjunto de datos con un comportamiento no lineal y la aproximación de la red neuronal con la función sigmoide a la salida como función de activación.

Como se menciona anteriormente en (2.5), se muestra que si se agregan capas sin una función de activación esto puede causar complicaciones e ineficiencia en el modelo, al agregar la función de activación esto se evita de manera adecuada, si se agrega la función de activación al termino de cada capa, (2.4) queda de la siguiente manera:

$$\vec{a} = f_3(f_2(f_1(\vec{x} \cdot W_1 + \vec{b}) \cdot W_2 + \vec{b}) \ldots \cdot W_k + \vec{b}) \tag{2.6}$$

Esto nos habilita para agregar las capas que se deseen utilizar o que sea necesario utilizar sin que éstas mismas hagan al modelo menos eficiente e inviable computacionalmente.

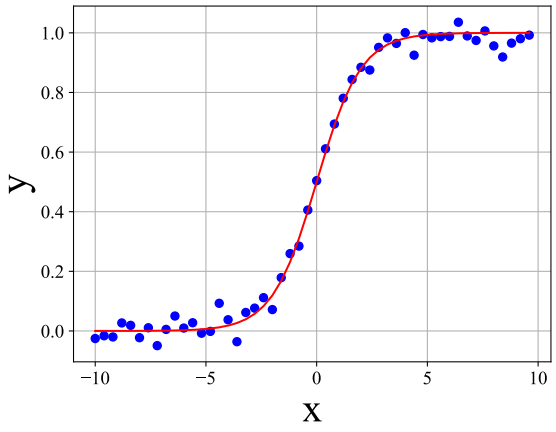


Figura 2.4: Datos NO lineales y una aproximación con la función sigmoide.

Funciones de activación

Existen muchas funciones de activación, de hecho, se podría decir que se puede utilizar cualquier función matemática existente una vez que se considere adecuada para la aproximación hacia los datos, pero esto también puede tener complicaciones que se mencionan en las secciones siguientes.

La Figura 2.5 muestra un perceptrón Multi-capa con dos capas ocultas, la primera con dos neuronas y la segunda con una neurona.

Algunas funciones de activación interesantes son :

- Función Sigmoide

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

- Función Tangente Hiperbólica (tanh)

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.8}$$

- Función ReLU (Rectified Linear Unit)

$$\text{ReLU}(x) = \text{máx}(0, x) \tag{2.9}$$

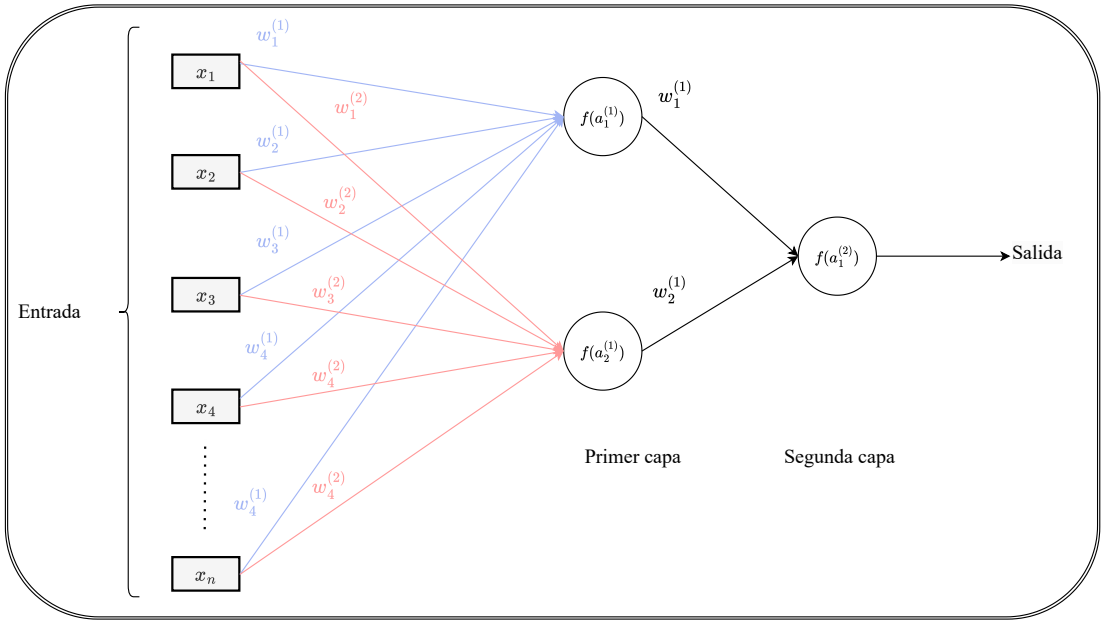


Figura 2.5: Perceptrón Multi-capa.

- Función Leaky ReLU

$$\text{Leaky ReLU}(x) = \begin{cases} \alpha x & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases} \tag{2.10}$$

donde α es un número pequeño positivo (por ejemplo, $\alpha = 0.01$).

- Función Exponential Linear Unit (ELU)

$$\text{ELU}(x) = \begin{cases} \alpha(e^x - 1) & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases} \tag{2.11}$$

donde α es un hiperparámetro que controla la saturación para valores negativos.

- Función Softplus

$$\text{Softplus}(x) = \log(1 + e^x) \tag{2.12}$$

- Función Swish

$$\text{Swish}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \tag{2.13}$$

- Función Softmax

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.14}$$

  donde x_i es el i-ésimo elemento del vector de entrada y n es el número de elementos en el vector de entrada.

En lo posterior se refiere a la función de activación como:

$$f_n(\vec{x}) \text{ ó } \vec{a} \tag{2.15}$$

Así, en este trabajo de tesis se utiliza la función de activación softmax (2.14) para generar una funcion de probabilidad a la salida de la red, y como resultado se obtenga la probabilidad de que lo que se introdujo a la entrada de la red partenesca a la clase n y ReLU (2.9) para evitar la linealidad en la red, donde la función de activación se selecciona por capas. En la Figura 2.5 se observan la Primer y Segunda capa, donde para este caso sencillo, se podría aplicar una función de activación distinta en cada capa, esto con la intención de que la aproximación hacia los datos sea mayor, cabe mencionar que si no se selecciona adecuadamente la función de activación, el modelo puede ser difícil de optimizar.

Una vez que se tienen todas las capas con las características propias previamente seleccionadas, es necesario optimizar el modelo. Para esto, se utiliza la función de pérdida que se detalla a continuación.

2.1.3. Función de Pérdida

Una vez que se tiene el modelo, es necesario entrenarlo o enseñarle mediante ejemplos. Para esto, es necesario el uso de una función de pérdida, esta función indica el error que existe entre lo que se predice o cataloga y lo que es la verdad o el “ground truth ” como lo llaman en inglés.

Al igual que existen diversas funciones de activación, también existen diversas funciones de error. Por lo que únicamente se mencionan un par para esta Sección. La más simple y conocida es mínimos cuadrados, que en un perceptrón se expresa de la siguiente manera:

$$f_{wb} = (a - y)^2 \tag{2.16}$$

donde a es la salida del perceptrón (2.15) o $f_n(\vec{x})$, y y corresponde a la etiqueta de la clase de entrada.

Una característica importante de las funciones de pérdida, es que tienen que ser convexas, ya que esta característica facilita su optimización y evita que al optimizar la función el algoritmo se estanque en mínimos locales. De esta manera, (2.16) es la que se tiene que minimizar, y al minimizarla, se garantiza que el error entre el valor que predice el perceptrón y los ejemplos es cercano a 0, es decir, que la aproximación hecha por el modelo es adecuada a los datos de entrenamiento.

La Figura 2.6 muestra gráficamente a (2.16) y sus curvas de nivel si únicamente se observa desde los ejes x y y, el encontrar el valor mínimo de esta función puede atraer otros problemas como el sobre ajuste hacia los datos.

Por otra parte, en la Ecuacion (2.17) corresponde a la función de pérdida crossentropy, la cual puede ser utilizada como un "indicador de error" que nos dice qué tan lejos está la predicción del modelo con respecto a la realidad, al comparar la probabilidad que el modelo asigna como categoría o clase correcta con la probabilidad real.

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i log(\hat{y}_i) \tag{2.17}$$

donde:

- y es un vector one-hot encoding de la clase real (un vector con N elementos, donde sólo un elemento es 1 y los demás son 0, indicando la clase verdadera).
- $\hat{y}$ es el vector de probabilidades predichas para las N clases.

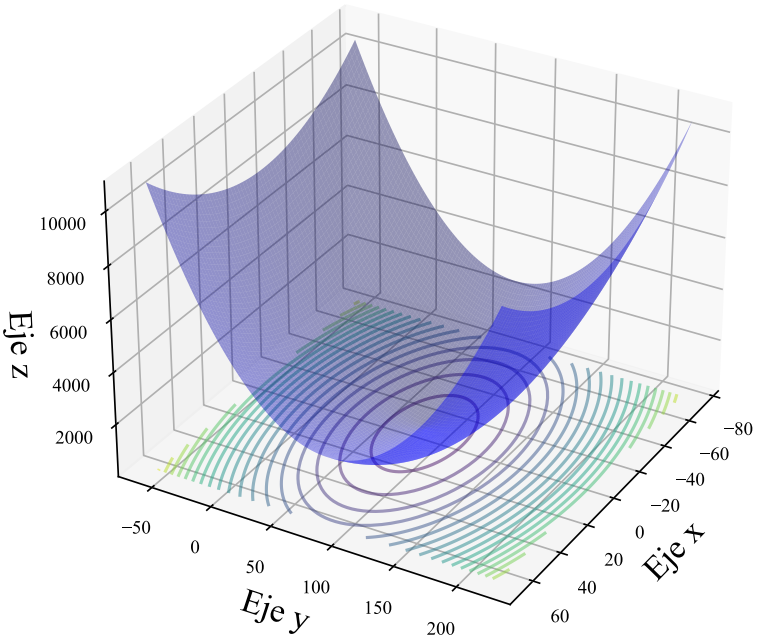


Figura 2.6: Mínimos cuadrados como función de pérdida.

- y_i es el valor en la posición i del vector y (1 corresponde a la clase real, 0 en otro caso).

- $\hat{y}_i$ es el valor en la posición i del vector de probabilidades predichas $\hat{y}$.

Esta función también es convexa por lo que es posible optimizar y llegar a un mínimo local mas cercano al global, el utilizar una función no convexa seria posible optimizarla sin embargo no se tendría ninguna garantía de que el mínimo al que que se llegue sea cercano o relativamente cercano a el mínimo global.

Una vez que se selecciona la función de pérdida que se va a utilizar en el entrenamiento del modelo, es necesario optimizar dicha función, lo cual se explica brevemente en la siguiente Sección. Para los fines de esta tesis, se utiliza la función en (2.17) como función

de pérdida o error.

2.1.4. Descenso de Gradiente

Descenso de gradiente es uno de los algoritmos mas populares de optimización y el más común en la optimización de redes neuronales. Actualmente, este algoritmo esta implementado en todas las librerías de aprendizaje profundo de última generación, por lo que en esta tesis se utiliza lo contenido en la libreria de acceso libre Tensor Flow [39].

Existen muchos algoritmos basados en descenso de gradiente [40], y la variante principal de cada una de estos algoritmos es la forma en la que se actualizan los parámetros de la función objetivo, que en un contexto de aprendizaje de maquina es la funcion de error que se muestra en (2.16), la cual se puede expresar como $f(\theta)$, donde los parámetros $\theta \in \mathbb{R}^d$ se actualizan en dirección opuesta del gradiente de la función objetivo $\nabla_\theta f(\theta)$ y la taza de aprendizaje η que indica el tamaño de paso que se toma para llegar a un mínimo. A continuación, se muestran algunas de las variantes.

Descenso de gradiente o descenso de gradiente por lote

Este algoritmo calcula el gradiente de la función de costo $f(\theta)$ de los parámetros θ para todo el conjunto de datos de entrenamiento [38]:

$$\theta = \theta - \eta \cdot \nabla_\theta f(\theta) \tag{2.18}$$

Como esta variante del algoritmo calcula el gradiente para todo el conjunto de datos para realizar una actualización, esto lo hace un algoritmo que puede ser muy lento y es inviable para conjuntos de datos que no caben en memoria. Esta variante del algoritmo garantiza llegar a un mínimo global para funciones convexas y a un mínimo local para funciones no convexas. Esta variante del algoritmo es muy estable.

Descenso de gradiente estocástico

Esta variante del algoritmo calcula el gradiente de una muestra y actualiza la función con ese gradiente, eso lo hace mucho más rápido [41].

$$\theta = \theta - \eta \cdot \nabla_\theta f(\theta; x^{(i)}; y^{(i)}) \tag{2.19}$$

En contraste con la variante de descenso de gradiente por lotes, que calcula gradientes redundantes e innecesarios, esta variante sólo se actualiza una vez en cada interación. La desventaja principal de esta variante, es que las actualizaciones en cada interación tienen una gran varianza, por lo que puede hacer que la función objetivo tenga muchas fluctuaciones. Por otro lado, esta variante habilita la posibilidad de brincar de un mínimo local a un potencial mejor mínimo local, pero puede complicar su convergencia hacia un miñimo exacto.

Descenso de gradiente por mini lotes

Esta variante aprovecha las ventajas mencionadas de las dos variantes anteriores al actualizar los parámetros con el gradiente calculado de un lote pequeño de el conjunto de datos [42], es decir:

$$\theta = \theta - \eta \cdot \nabla_\theta f(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.20}$$

De esta manera se reduce la varianza al actualizar los valores, y de esta manera se vuelve relativamente rápido.

La Figura 2.6 muestra una función de pérdida en 3 dimensiones, que en un conjunto de datos es el numero de características, y sus curvas de nivel en los ejes x y y. La Figura 2.7 muestra una comparativa del comportamiento de los 3 algoritmos mencionados anteriormente al buscar el valor óptimo para la función de pérdida (2.16), que corresponde a mínimos cuadrados. Se pueden observar las ventajas y desventajas que se mencionan anteriormente.

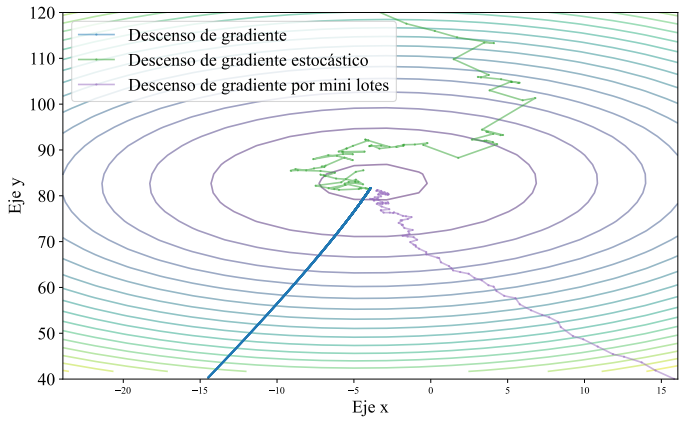


Figura 2.7: Descenso de gradiente.

Estos algoritmos tienen algunos desafíos que son complicados de resolver, tales como: escoger una taza de aprendizaje apropiada, intentar ajustar la taza de aprendizaje durante el entrenamiento, actualizar parcialmente el modelo, evitar quedar atrapado en no óptimos mínimos locales, entre otros.

Para resolver estos desafíos, se han propuesto los siguientes algoritmos de optimización basados en el descenso del gradiente, los cuales se presentan como: Momentun [43], Adagrad [44], Adadelta [45], RMSprop [46], Adam [47], Nadam [48].

En esta tesis se utiliza el algoritmo Adam [47], el cual utiliza las ventajas de los algoritmos RMSprop [46] y Adagrad [44], respectivamente, y se encuentra en la librería [39] de código abierto.

Lo más sobresaliente del algoritmo Adam, es que calcula promedios móviles del primer momento (la media) y del segundo momento (la varianza no centralizada) de los gradientes, es decir:

Primer y segundo momento:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{2.21}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2.22}$$

donde g_t es el gradiente en el tiempo t, m_t es el promedio móvil del primer momento y v_t es el promedio móvil del segundo momento. Al inicializar m_t y v_t se observa que se crea un sesgo hacia cero, especialmente cuando las tazas decaen hacia cero, para corregir esto, el primer y segundo momento se estiman como:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.23}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.24}$$

de tal forma que la actualización de θ queda de la siguiente manera:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \tag{2.25}$$

La ecuación (2.25) muestra el algoritmo que se utiliza para el entrenamiento del modelo de red neuronal propuesto en esta tesis.

2.2. Técnicas de Aprendizaje Profundo

Durante el desarrollo de la inteligencia artificial se han creado numerosas técnicas de aprendizaje profundo y arquitecturas de redes neuronales tales como las siguientes.

- Redes Neuronales Clásicas.
- Redes Neuronales Convolucionales.
- Redes Neuronales Recurrentes.
- Máquinas Restrictivas de Boltzmann.
- Redes Generativas de Confrontación (GANs).
- Mapas Auto-organizados.
- Aprendizaje Profundo por Refuerzo.

- Auto-encoders.
- Backpropagation.
- Aprendizaje por Transferencia.
- Sintonización Fina.

Esta Sección describe únicamente tres técnicas de lo anteriormente mencionado, las cuales son: (*i*) redes neuronales convolucionales, (*ii*) aprendizaje por transferencia y (*iii*) sintonización fina, las cuales son parte importante de esta tesis.

2.2.1. Redes Neuronales Convolucionales

Este tipo de red neuronal se compone de capas densas únicamente. Así, dentro de estas capas, esta tesis utiliza dos capas sumamente importantes, las cuales son: capa convolucional y capa de agrupamiento. Aunque no se detallan todas sus propiedades, contienen dos propiedades muy valiosas dentro del análisis de señales y son: linealidad e invarianza a traslación.

El fundamento matemático de las redes neuronales convolucionales es muy similar al de las redes neuronales clásicas o densas, que se mencionan anteriormente, la diferencia más grande recae en el operador que se utiliza en las capas internas de la red, específicamente, entre el vector de pesos $\vec{w}$ y el vector de características $\vec{x}$, ya que en lugar de utilizar el operador producto punto entre los dos vectores, se utiliza el operador convolución entre una imagen y/o una señal en una dimensión, y el kernel o filtro. Otra diferencia es que a la salida, en lugar de tener un escalar, se tiene una imagen o señal, dependiendo de lo que se utiliza en la entrada, por lo que, si se utilizan N neuronas, se obtiene una señal con N canales en lugar de un vector de tamaño N.

Algunas características importantes que se obtienen en la red, es que ahora la red no está totalmente conectada, es decir, cada característica en el vector $\vec{x}$ no se asocia directamente a alguno de los componentes del vector de pesos $\vec{w}$, si no que el vector $\vec{x}$

se asocia con el vector de pesos $\vec{w}$ sin importar si son del mismo tamaño. Es decir, esta asociada la señal contenida en el vector $\vec{x}$ con lo que mencionamos anteriormente, por lo que se llama kernel o filtro de pesos $\vec{w}$ y no una asociación elemento a elemento. Esto permite que el $\vec{w}$ pueda ser de un tamaño diferente al del $\vec{x}$.

En este Sección, se describen brevemente algunas de las capas utilizadas para la creación de redes neuronales convolucionales y más particularmente, las capas que se utilizan en esta tesis, las cuales son: capa con el operador convolución o capa convolucional, capa de pooling o agrupamiento, capa de normalización de lotes o batch normalization, capa de abandono o dropout, capa de aplanado.

Capa con el operador convolución o capa convolucional

En matemáticas, y en particular en el análisis de señales, una convolución es un operador matemático que transforma dos funciones f y g en una tercera función, que en cierto sentido, representa la magnitud en la que se superponen f a una versión trasladada e invertida de g.

Para funciones discretas, la convolución se puede expresar de la siguiente manera:

$$f[m] * g[m] = \sum_n f[n] \cdot g[m-n] \tag{2.26}$$

donde $*$ indica el operador convolución.

La Figura 2.8 muestra un ejemplo simple de la convolución entre una señal y un kernel.

En esta convolución al centrar el kernel en 0 los elementos $g[m-n]$ que no existen se consideran 0 para mantener el tamaño de la señal, y no se agregan elementos adicionales a la señal. De esta forma, es como utiliza el operador convolución a lo largo de esta tesis, existen otras formas de realizar el operador como es con padding (agregar ceros a la señal), aplicando convolución circular, entre otras que no se mencionan aquí pero son comunes [49].

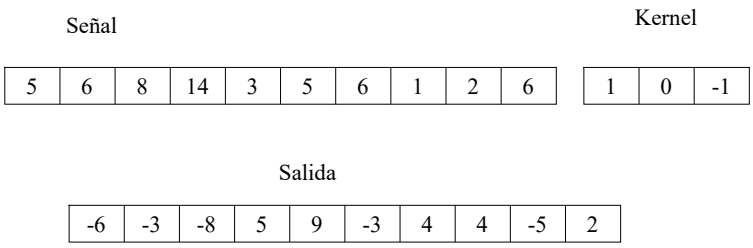


Figura 2.8: Convolución.

La idea principal de esta capa, es reemplazar el producto punto de la expresión $\vec{a} = \vec{x} \cdot \mathbf{w} + \vec{b}$ por el operador convolución, de tal forma que la salida del perceptrón sea:

$$\vec{a} = \vec{x} * \mathbf{w} + \vec{b} \tag{2.27}$$

que al agregar una función de activación, la salida de la capa es:

$$x_o^c = f(\vec{x} * \mathbf{w} + \vec{b}) \tag{2.28}$$

Es decir, que básicamente la estructura del modelo en cuanto a variables y el orden de dichas variables es el mismo, la diferencia principal radica en el operador, ya que en lugar de utilizar el operador producto punto '$\cdot$', se utiliza el operador convolución '$*$'.

Capa de pooling o agrupamiento

El objetivo principal del pooling o agrupamiento es reducir la dimensionalidad de la señal manteniendo la mayor cantidad de información posible.

El pooling o agrupamiento es una función que remplaza la salida de la red neuronal a una ubicación determinada con una estadística resumida de las salidas cercanas.

La Figura 2.9 muestra el pooling en una ventana de tamaño 2.

La operación de agrupación o pooling informa la salida dentro de una vecindad [50], la agrupación puede informar el valor máximo, el promedio, el valor mínimo o algo interesante contenido en la vecindad, en el caso anterior no informa nada en específico Por

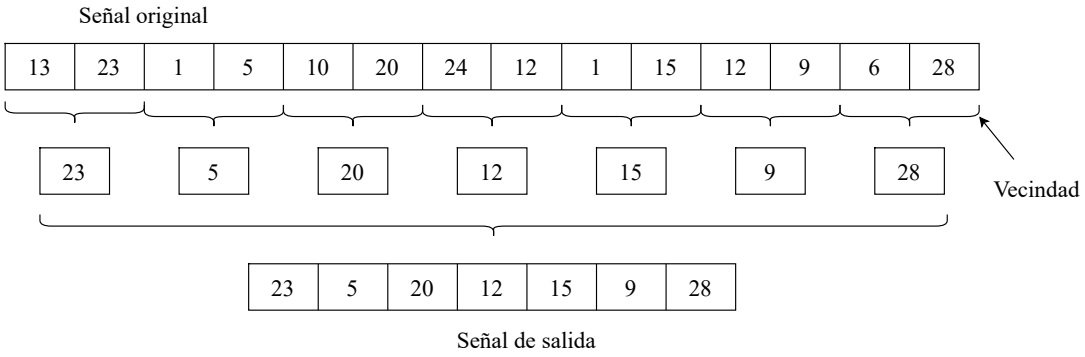


Figura 2.9: Agrupamiento o pooling.

otro lado, la Figura 2.10 muestra la función de agrupación máxima o *max pooling* con una ventana de tamaño 2.

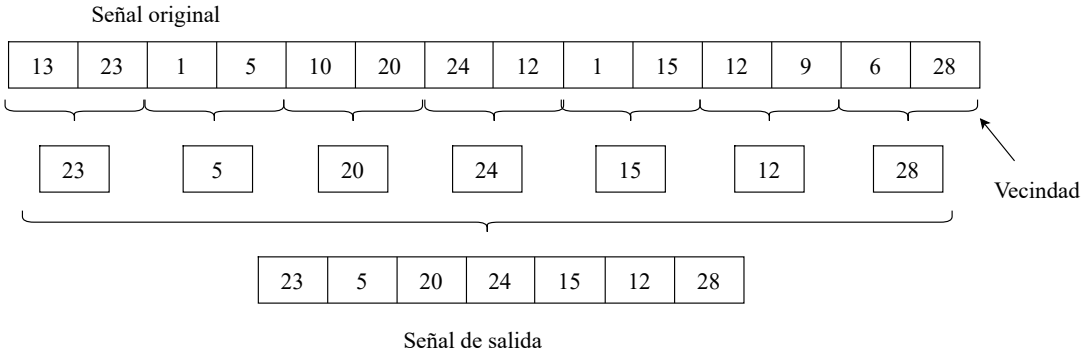


Figura 2.10: Agrupamiento máximo.

Otras funciones de agrupamiento que se puede aplicar a la vecindad o ventana son: promedio, norma, promedio ponderado, entre otras.

En todos los casos, la agrupación ayuda a que la representación se vuelva aproximadamente invariante ante pequeñas traslaciones de la entrada. La invarianza a traslación significa que si trasladamos la entrada en una pequeña cantidad, los valores de la mayoría de las salidas agrupadas no cambian.

La idea de una capa de agrupación o una capa de pooling es que esta capa ayude a reducir las dimensiones de los datos de entrada conservando al mismo tiempo la información más importante.

Capa de normalización de lotes o batch normalization

La normalización por lotes [51], es una capa utilizada en redes neuronales para mejorar la velocidad, rendimiento y estabilidad de la red en la etapa de entrenamiento. Para esto, se inserta en una red neuronal para normalizar las salidas de la capa anterior, lo cual se realiza por lotes de datos durante el entrenamiento. La normalización se realiza a cada dato que pasa a través de esta capa y se realiza como se describe a continuación.

Se utiliza la media y la varianza para normalizar las activaciones:

$$\hat{x} = \frac{x - \mu B}{\sqrt{\alpha_B^2 + \epsilon}} \tag{2.29}$$

donde x es la activación original y ϵ es un pequeño valor para evitar divisiones por cero. Después de la normalización, se aplica una transformación lineal mediante (2.30), que permite al modelo aprender la escala y los desplazamientos óptimos.

$$y = \gamma\hat{x} + \beta \tag{2.30}$$

donde γ y β son los parámetros que el modelo aprende.

Algunas de las ventajas que esta capa brinda al modelo durante el entrenamiento son:

- Reduce el problema de desvanecimiento de gradientes, es decir, que los gradientes se hagan muy pequeños.
- Ayuda a mitigar la sensibilidad en la inicialización de los pesos, permitiendo una convergencia más rápida y estable.
- Ayuda a mantener las salidas de las capas ocultas dentro de un rango controlado, lo que estabiliza el proceso de entrenamiento.

Capa de abandono o dropout

Es una capa de regularización [52, 53] que ayuda a prevenir el sobre-ajuste hacia los datos. Es un método que aleatoriamente "apagan" un porcentaje de neuronas en una capa durante cada paso de entrenamiento, lo que significa que esas neuronas no participan en la propagación hacia adelante ni en la propagación hacia atrás en ese paso específico, es decir, esas neuronas no participan en el entrenamiento. Sin embargo, durante la fase de prueba (inferencia), todas las neuronas están activas, pero sus salidas se escalan para mantener la magnitud de la salida de las capas.

Durante la etapa de entrenamiento, la capa de abandono actúa de la siguiente manera:

- Para cada paso de entrenamiento, se selecciona aleatoriamente un subconjunto de neuronas que se "apagarán" con una probabilidad p.
- Las neuronas no apagadas se escalan por $\frac{1}{1-p}$, para mantener la magnitud esperada en la salida de las capas.

Mientras que durante la etapa de inferencia o predicción no se apaga ni se escala ninguna neurona.

Capa de Aplanado

La capa de aplanado toma una entrada que puede ser una matriz multidimensional (por ejemplo, una imagen con dimensiones de altura, anchura y canales de color) o para el próposito de esta tesis, un vector multidimensional que los convierte en un vector unidimensional, ya que las capas densas requieren un vector unidimensional como entrada.

La capa de aplanado o "flatten" en inglés es crucial, porque permite que las características extraídas por una o más capas convolucionales y de agrupamiento, sean utilizadas por las capas densas, que son las responsables de hacer la clasificación final. Sin

la capa de aplanado, no es posible conectar directamente la salida de una capa convolucional a una capa densa, ya que la dimensionalidad no sería la misma.

En resumen, la capa de aplanamiento es una herramienta esencial en el diseño de arquitecturas de redes neuronales, especialmente, para conectar capas convolucionales con capas densas, permitiendo una transición fluida entre diferentes tipos de capas.

Así, las capas mencionadas anteriormente constituyen una red neuronal convolucional, una vez que se entrena la red neuronal convolucional y ya ha aprendido algo, es posible transferir ese conocimiento hacia otra red u otro modelo.

2.2.2. Aprendizaje por Transferencia

El aprendizaje por transferencia en el campo del aprendizaje de máquina, se refiere a aprovechar el conocimiento adquirido durante la resolución de una tarea para mejorar el rendimiento en otra tarea relacionada pero distinta. En lugar de comenzar el aprendizaje desde cero para cada tarea, se transfieren los conocimientos previamente aprendidos de una tarea fuente (donde se dispone de datos o experiencia) a una tarea objetivo (donde se quiere mejorar el rendimiento).

Por ejemplo, si existe una red neuronal que puede clasificar sí en una imagen existe un mapache u otros animales como se muestra en la Figura 2.11, esta misma red es muy útil si ahora, en lugar de detectar o clasificar mapaches, queremos clasificar coatíes o martuchas como se muestra en la Figura 2.12, ya que las características aprendidas por el modelo se pueden aprovechar [54, 55, 56, 57, 58], por lo tanto, no es necesario entrenar al nuevo modelo desde cero.

Esto es extraordinariamente útil si ocurren 2 factores:

- El conjunto de datos de coatíes o martuchas es reducido.
- El tiempo para entrenar un nuevo modelo es reducido.

Al utilizar esta técnica llamada transferencia de aprendizaje, aprovechamos el conocimiento

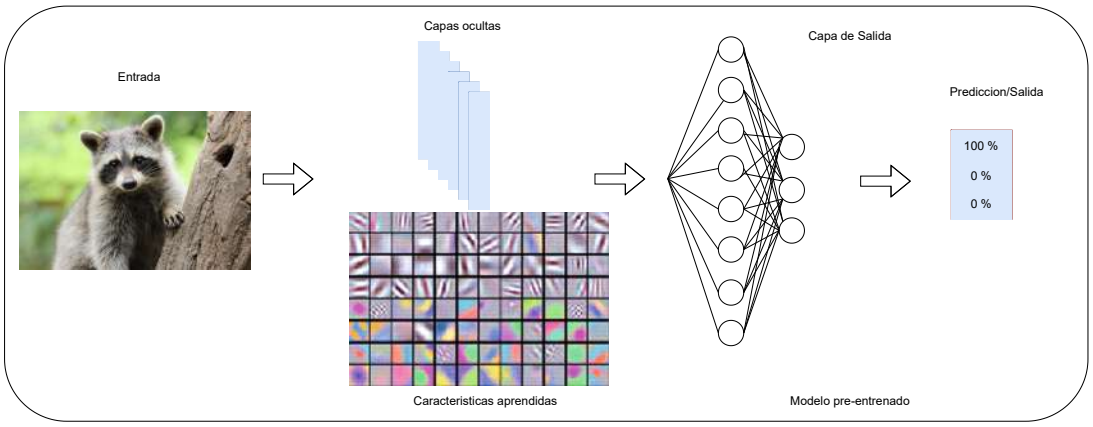


Figura 2.11: Modelo Pre-entrenado.

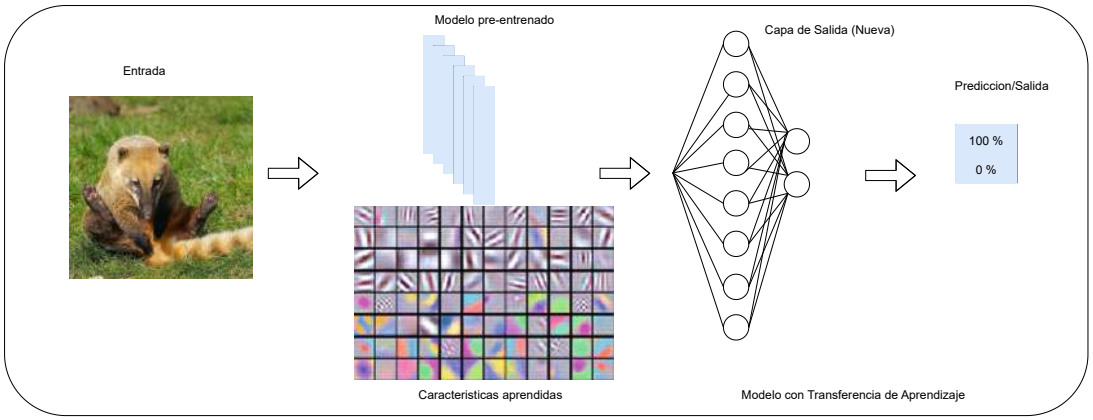


Figura 2.12: Modelo con Transferencia de Aprendizaje.

adquirido por un modelo previamente entrenado y lo adaptamos a las necesidades de la tarea que deseamos desempeñar, de esta manera, es posible obtener resultados aceptables o mejorar los resultados que se adquieren con un conjunto de datos reducido. También, el tiempo de entrenamiento traducido a costo computacional se reduce considerablemente. Así, esta técnica se aplica en aprendizaje supervisado y existen consideraciones que se deben tomar para utilizar esta técnica como el congelamiento de las capas, para no perder el conocimiento ya adquirido entre otras.

2.2.3. Sintonización Fina

La sintonización fina [59], también conocida como "fine-tuning" en inglés, implica ajustar un modelo pre-entrenado en una tarea específica, en lugar de entrenar el modelo desde cero. Esta técnica esta ligada a la tecnica de aprendizaje por transferencia, podría decirse que es el paso siguiente de realizar la transferencia de aprendizaje, la diferencia recae en que en la etapa de sintonización fina las capas no están congeladas, y es particularmente útil cuando se dispone de un modelo pre-entrenado en una tarea relacionada y se desea adaptar ese modelo a una nueva tarea o conjunto de datos específicos.

Para la ejemplificación de esta técnica, es mas fácil utilizar una función en tres dimensiones. La Figura 2.13 muestra las curvas de nivel de la Figura 2.6 de un conjunto de datos con valores para x y y, así como la solución a la que llega el algoritmo de descenso de gradiente por mini lotes, al ser entrenado con un conjunto de datos x y y similar pero no igual.

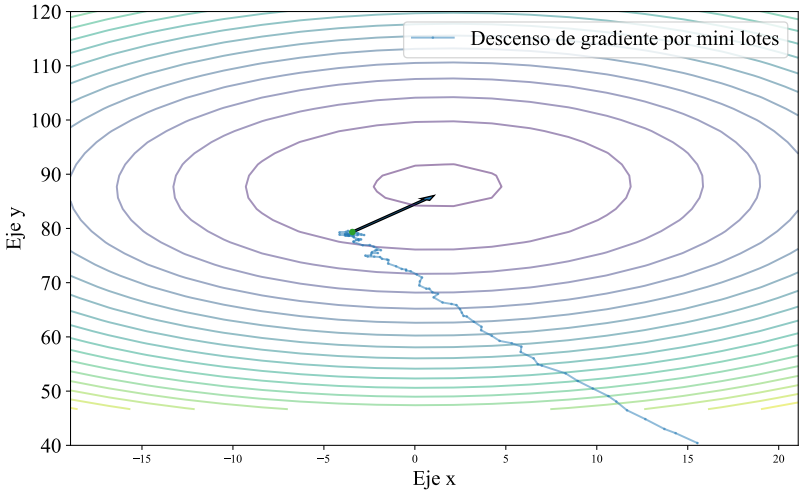


Figura 2.13: Sintonización fina.

La idea es utilizar el conocimiento ya aprendido, que en otras palabras, es la solución a la que llega y busca una solución óptima para el conjunto de datos nuevo. Para

utilizar esta técnica y aprovechando que el modelo ya está muy cerca a una solución y no se quiere perder esa solución, se reduce la taza de aprendizaje o "learning rate" a un valor muy pequeño, por lo que se vuelve a entrenar por un periodo de tiempo relativamente corto.

En la Figura 2.13, la linea azul representa la solución a la que llega un modelo hipotético, mientras que las curvas de nivel muestran una nueva función, es decir, un nuevo problema. Así, la solución a la que llega el modelo anterior ya esta muy cerca de la nueva solución requerida por este nuevo problema, por lo que no es necesario entrenarlo desde cero, incluso se puede hacer con un tamaño de paso pequeño y un conjunto de datos menor al que se utilizó anteriormente, esta es la sintonización fina, utilizar el conocimiento adquirido en un modelo para solucionar un problema diferente.

Esta técnica está muy relacionada con la técnica de aprendizaje por transferencia, de hecho, la sintonización fina no se puede llevar a cabo sin la técnica de aprendizaje por transferencia.

2.3. Métricas

En el contexto del aprendizaje automático, una métrica es una medida cuantitativa utilizada para evaluar el rendimiento de un modelo. Las métricas son esenciales porque proporcionan una forma objetiva de comparar y contrastar diferentes modelos o configuraciones del mismo modelo. Las métricas permiten determinar el funcionamiento de un modelo en relación con el problema específico que están tratando de resolver.

Las métricas varían ampliamente dependiendo del problema y el objetivo específico. A continuación, se describen algunas categorías y ejemplos de métricas comunes que se pueden relacionar a esta tesis:

- Precisión (accuracy): Es la proporción de predicciones correctas sobre el total de

predicciones.

$$\text{Precisión} = \frac{\text{Número de predicciones correctas}}{\text{Total de predicciones}} \tag{2.31}$$

- Precisión (precision): Se refiere a la proporción de verdaderos positivos sobre el total de predicciones positivas.

$$\text{Precisión} = \frac{\text{TP}}{\text{TP+FP}} \tag{2.32}$$

donde TP significa positivo verdadero por sus siglas del inglés (True Positive) y FP significa falso positivo (del inglés False Positive).

- Exhaustividad (recall): Es la proporción de verdaderos positivos sobre el total de casos positivos reales.

$$\text{Exhaustividad } = \frac{\text{TP}}{\text{TP+FN}} \tag{2.33}$$

donde FN significa falso negativo.

- $f1$ score: Es la media armónica de precisión y exhaustividad y se define como:

$$f1 = 2 \cdot \frac{\text{Exhaustividad} \cdot \text{Precisión}}{\text{Exhaustividad} + \text{Precisión}} \tag{2.34}$$

Las métricas proporcionan una forma cuantitativa de evaluar el rendimiento de un modelo, lo que es crucial para determinar si el modelo está cumpliendo con los objetivos esperados.

En general, una vez que se tienen las técnicas de aprendizaje, las métricas y todo el fundamento matemático, las redes neuronales no son útiles si no tienen un extenso conjunto de datos de entrenamiento, es decir, que la calidad y eficiencia de estos modelos están directamente relacionadas con los datos de entrenamiento y de validación, por lo que en el siguiente Capítulo se introduce una herramienta útil y fiable para la generación de conjuntos de datos de eventos de calidad de la energía.

Capítulo 3

PQ-SyDa: Generador de conjunto de datos de eventos de calidad de la energía

3.1. Descripción de la caja de herramientas

En la era de los modelos basados en datos, los modelos de aprendizaje automático (ML) predominan en la literatura de investigación tecnológica y de ingeniería. A pesar de la precisión y el uso generalizado de los modelos basados en ML, su calidad depende en gran medida de los datos utilizados para ajustar el modelo. Por lo tanto, es fundamental tener en cuenta datos representativos y precisos que describan correctamente los fenómenos o el sistema que se va a aprender.

Hasta donde sabemos, existen algunos conjuntos de datos de carácter público estandarizados sobre eventos de calidad de energía (PQ). Además, la mayoría de ellos necesitan ser etiquetados y solo consideran un conjunto limitado de eventos PQ, además de que necesitan más detalles sobre su frecuencia de muestreo y sus valores pico. Entonces, las señales registradas durante eventos reales suelen ser difíciles de encontrar y solo consideran

eventos simples que deben analizarse antes de que puedan usarse para ajustar un modelo de ML. Este hecho dificulta la comparación de resultados entre diferentes estudios realizados por la comunidad científica. Una alternativa es una estrategia de aumento de datos que utiliza señales modeladas matemáticamente y que se pueden implementar para contar con suficientes señales de eventos PQ.

El toolbox que se presenta en esta tesis ayuda en la generación de conjuntos de datos estandarizados para evaluar enfoques basados en datos para modelar eventos PQ relacionados a los sistemas de potencia. Los datos sintéticos se pueden integrar con eventos reales para construir un conjunto de datos lo suficientemente grande y adecuado para su uso con enfoques de aprendizaje profundo. La herramienta propuesta puede crear conjuntos de datos de señales sintéticas para cualquier combinación de 29 eventos PQ diferentes, donde todos los eventos se etiquetan automáticamente. Así, el usuario puede definir todo el muestreo y los valores relacionados con la senãl cuando se crea la base de datos. La exactitud de las bases de datos generadas se garantiza gracias a la herramienta implementada, que se basa en la teoría detrás del modelo matemático integral presentado en la literatura de PQ. Aunque este trabajo se superpone con el de [60], la propuesta va más allá, ya que se proporciona una interfaz gráfica amigable, además de que muchos parámetros del conjunto de datos pueden ser definidos por el usuario y se realiza un punto de referencia para la clasificación de eventos PQ para proporcionar un punto de referencia a los usuarios interesados. Además, el sistema implementado en [60] está más orientado al modelado de sistemas, mientras que el presente trabajo se centra en la estandarización de datos para tareas de clasificación o detección.

3.1.1. Características clave

Las principales características clave se resumen de la siguiente manera:

- **Generación de datos sintéticos**: permite la creación de datos artificiales basados en eventos PQ predefinidos. Esto facilita la creación de conjuntos de datos persona-

lizados con características específicas.

- **Control de dimensionalidad**: proporciona opciones para ajustar la dimensionalidad de los datos, lo que permite realizar pruebas integrales de algoritmos. Esto se logra modificando la cantidad de ciclos y muestras por ciclo.
- **Etiquetado automático**: este toolbox ofrece capacidades para etiquetar automáticamente los conjuntos de datos generados según reglas predefinidas, lo que agiliza el proceso de preparación de datos para el entrenamiento y la validación de modelos.
- **Integración con estrategias de ML**: compatible con las principales estrategias de ML como TensorFlow ([61]), PyTorch ([62]), scikit-learn ([63]), entre otros; lo que facilita la integración perfecta de los conjuntos de datos generados en los procesos de desarrollo y experimentación.
- **Visualización y análisis exploratorio básico**: esta plataforma incluye herramientas para visualizar y realizar análisis exploratorios de los conjuntos de datos generados, lo que permite una comprensión profunda de las características y distribuciones de los datos antes de usarlos en modelos de ML.

En resumen, este toolbox permite generar y visualizar hasta 29 eventos de calidad de energía y modificar sus características. Así, como parte de esta tesis, en este Capítulo se presenta el toolbox de interfaz gráfica de usuario (GUI, del inglés Graphical User Interface) desarrollada en Python y muy fácil de usar, con el objetivo de generar de manera sintética conjuntos de datos de eventos de calidad de energía. Esta GUI puede proporcionar y exportar una colección versátil de herramientas diseñadas específicamente para crear conjuntos de datos robustos y realistas para su uso en la evaluación y prueba de algoritmos de aprendizaje automático y aprendizaje profundo para eventos de calidad de energía que pueden presentarse en los sistemas de elétricos de potencia. Por lo que, se adopta una implementación sencilla para generar conjuntos de datos con hasta 29 eventos sintéticos diferentes de calidad de energía con opciones de etiquetas automatizadas [64, 65].

3.2. Modelos matemáticos para eventos de calidad de la energía

Las representaciones matemáticas de las perturbaciones más comunes para eventos de calidad de energía se muestran en la Tabla 3.1 [60, 66]. Nótese que se incluyen perturbaciones de calidad de energía simples y combinadas, es decir, eventos simples, dobles, triples y cuádruples.

Así, la Tabla 3.1 muestra los modelos matemáticos correspondientes a los 29 eventos PQ que se utilizan en esta tesis. Nótese que existen ciertas variables para representar cada uno de estos eventos, por lo que, se aplica un conjunto de parámetros aleatorios para los modelos PQ con el fin de generar un comportamiento más realista de los fenómenos generados, y de esta forma, crear el conjunto de datos de perturbaciones PQ sintéticas para aplicaciones de aprendizaje automático y profundo.

De este modo, los parámetros comunes para todas las perturbaciones PQ presentadas en la Tabla 3.1 se muestran en (3.1), donde $w = 2\pi f$ y $A = 1$.

$$\begin{aligned} &-\pi \leq \phi \leq \pi \\ &u(t) = \begin{cases} 0 & : t < 0 \\ 0 & : t \geq 0 \end{cases} \end{aligned} \tag{3.1}$$

Ahora, los parámetros comunes para todas las perturbaciones que incluyen sags, swells e interrupciónes de la Tabla 3.1, se presentan en (3.2).

$$\begin{aligned} &T \leq t_2 - t_1 \leq (N-1)T \\ &0.1 \leq \alpha \leq 0.9;\ 0.1 \leq \beta \leq 0.8;\ 0.9 \leq \rho \leq 1.0 \end{aligned} \tag{3.2}$$

Ahora, para los parámetros comunes que involucran perturbaciones transitorias en la Tabla 3.1, éstos se representan en (3.3).

$$\begin{aligned} &0.222 \leq \psi \leq 1.11 \\ &T \leq t_q \leq (N-1)T \\ &t_b = t_a + 1ms \end{aligned} \tag{3.3}$$

Tabla 3.1: Representación Matemática de los Eventos PQ.

Label	Event	Model
0	Señal Pura	$v(t) = Asin(wt - \phi)$
1	Sag	$v(t) = A(1 - \alpha(u(t - t_1) - u(t - t_2)))sin(wt - \phi)$
2	Swell	$v(t) = A(1 + \beta(u(t - t_1) - u(t - t_2)))sin(wt - \phi)$
3	Interrupcion	$v(t) = v(t) = A(1 - \rho(u(t - t_1) - u(t - t_2)))sin(wt - \phi)$
4	Spike/ Impulso	$v(t) = A[sin(wt - \phi) - \Psi(e^{-750(t-t_a)} - e^{-344(t-t_a)})(u(t - t_a) - u(t - t_b))]$
5	Transitorio Oscilatorio	$v(t) = A[sin(wt - \phi) + \beta e^{t-t_I/\tau} sin(w_n(t - t_I) - \vartheta)((u(t - t_{II}) - u(t - t_I)))]$
6	Ármonicos	$v(t) = A[sin(wt - \phi) + \sum_{n=3}^{7} \alpha_n sin(nwt - \vartheta_n)]$
7	Ármonicos + Sag	$v(t) = A(1 - \alpha(u(t - t_1) - u(t - t_2)))[sin(wt - \phi) + \sum_{n'=3}^{7} \alpha_{n'} sin(n'wt - \vartheta_{n'})]$
8	Ármonicos + Swell	$v(t) = A(1 + \beta(u(t - t_1) - u(t - t_2)))[sin(wt - \phi) + \sum_{n'=3}^{7} \alpha_{n'} sin(n'wt - \vartheta_{n'})]$
9	Flicker	$v(t) = A[1 + \lambda sin(w_f t)]sin(wt - \phi)$
10	Flicker + Sag	$v(t) = A[1 + \lambda sin(w_f t) - \alpha(u(t - t_1) - u(t - t_2))]sin(wt - \phi)$
11	Flicker + Swell	$v(t) = A[1 + \lambda sin(w_f t) + \beta(u(t - t_1) - u(t - t_2))]sin(wt - \phi)$
12	Sag + Trans. Osc.	$v(t) = A[sin(wt - \phi)(1 - \alpha(u(t - t_1) - u(t - t_2)))+$ $\beta e^{-(t-t_{I'})/\tau} sin(w_n(t - t_{I'}) - \vartheta)(u(t - t_{II'}) - u(t - t_{I'}))]$
13	Swell Trans. Osc.	$v(t) = A[sin(wt - \phi)(1 - \alpha(u(t - t_1) - u(t - t_2)))+$ $\beta e^{-(t-t_{I'})/\tau} sin(w_n(t - t_{I'}) - \vartheta)(u(t - t_{II'}) - u(t - t_{I'}))]$
14	Sag + Ármonicos	$v(t) = A[sin(wt - \phi_1) + (-\alpha(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n'wt - \vartheta_{n''})]$
15	Swell + Ármonicos	$v(t) = A[sin(wt - \phi_1) + \beta(u(t - t_1) - u(t - t_2)) \sum_{n''=1}^{5} \alpha_{n''} sin(n'wt - \vartheta_{n''})]$
16	Notch	$v(t) = A[sin(wt - \phi) - sign(sin(wt - \phi)) \sum_{n=0}^{Nc-1} k(u(t - (t_c + sn))u(t - (t_d + sn)))]$
17	Ármonicos + Sag + Flicker	$v(t) = A(1 + \lambda sin(w_f t))[sin(wt - \phi) + \sum_{n'=3}^{5} \alpha_{n'} sin(n'wt - \vartheta_{n'})](1 - \alpha(u(t - t_1) - u(t - t_2)))$
18	Ármonicos + Swell + Flicker	$v(t) = A(1 + \lambda sin(w_f t))[sin(wt - \phi) + \sum_{n'=3}^{5} \alpha_{n'} sin(n'wt - \vartheta_{n'})](1 + \beta(u(t - t_1) - u(t - t_2)))$
19	Sag + Ármonicos + Flicker	$v(t) = A[sin(wt - \vartheta_1) + (1 + \lambda sin(w_f t))(-\alpha(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n''wt - \vartheta_{n''})]$
20	Swell + Ármonicos + Flicker	$v(t) = A[sin(wt - \vartheta_1) + (1 + \lambda sin(w_f t))(\beta(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n''wt - \vartheta_{n''})]$
21	Sag + Ármonicos + Osc.	$v(t) = A[sin(wt - \phi)(-\alpha(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_{I'})/\tau} sin(w_n(t - t_{I'}) - \vartheta)(u(t - t_{II'}) - u(t - t_{I'}))]$
22	Swell + Ármonicos + Osc.	$v(t) = A[sin(wt - \phi)(\beta(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_{I'})/\tau} sin(w_n(t - t_{I'}) - \vartheta)(u(t - t_{II'}) - u(t - t_{I'}))]$
23	Ármonicos + Sag + Osc.	$v(t) = A[(1 - \alpha(u(t - t_1) - u(t - t_2))) \sum_{n''+1}^{5} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_I)/\tau} sin(w_n(t - t_I) - \vartheta)(u(t - t_{II} - u(t - t_I)))]$
24	Ármonicos + Swell + Osc	$v(t) = A[(1 + \beta(u(t - t_1) - u(t - t_2))) \sum_{n''+1}^{5} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_I)/\tau} sin(w_n(t - t_I) - \vartheta)(u(t - t_{II} - u(t - t_I)))]$
25	Ármonicos + Sag + Flicker + Osc.	$v(t) = A[(1 - \alpha(u(t - t_1) - u(t - t_2))) \sum_{n''+1}^{5} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_I)/\tau} sin(w_n(t - t_I) - \vartheta)(u(t - t_{II} - u(t - t_I)))](1 + \lambda sin(w_f t))$
26	Ármonicos + Swell + Flicker + Oscillatorio	$v(t) = A[(1 + \beta(u(t - t_1) - u(t - t_2))) \sum_{n''+1}^{5} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_I)/\tau} sin(w_n(t - t_I) - \vartheta)(u(t - t_{II} - u(t - t_I)))](1 + \lambda sin(w_f t))$
27	Sag + Ármonicos + Flicker + Osc.	$v(t) = A[sin(wt - \vartheta_1) + (-\alpha(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_{I'})/\tau} sin(w_n(t - t_{I'}) - \vartheta)(u(t - t_{II'}) - u(t - t_{I'}))(1 + \lambda sin(w_f t))]$
28	Swell + Ármonicos + Flicker + Osc.	$v(t) = A[sin(wt - \vartheta_1) + (\beta(u(t - t_1) - u(t - t_2))) \sum_{n''=1}^{5} \alpha_{n''} sin(n''wt - \vartheta_{n''})+$ $\beta e^{-(t-t_{I'})/\tau} sin(w_n(t - t_{I'}) - \vartheta)(u(t - t_{II'}) - u(t - t_{I'}))(1 + \lambda sin(w_f t))]$

Para los parámetros comunes que contienen perturbaciones de flicker, se expresan en (3.4), donde $w_f = 2\pi f_f$.

$$\begin{aligned} 0.05 \leq \lambda \leq 0.1 \\ 8 \leq f_f \leq 25Hz \end{aligned} \tag{3.4}$$

Los parámetros comunes para perturbaciones relacionadas con transitorios oscilatorios se muestran en (3.5), con $w_n = 2\pi f_n$.

$$\begin{aligned} &300 \leq f_n \leq 900Hz;\ 8ms \leq \tau \leq 40ms;\ -\pi \leq \vartheta \leq \pi \\ &0.5T \leq t_{II} - t_I \leq \frac{N}{3.33}T \\ &\frac{T}{5} \leq t_{II'} - t_{I'} \leq t_2 - t_1; t_{I'} \geq t_1;\ t_{II'} \leq t_2 \end{aligned} \tag{3.5}$$

Similarmente, los parámetros comunes para perturbaciones relacionadas con armónicos se muestran en (3.5) con $w_n = 2\pi f_n$.

$$\begin{aligned} &0.05 \leq \alpha_n \leq 0.15; -\pi \leq \vartheta_n, \vartheta_{n'}, \vartheta_{n''} \leq \pi \\ &n' = \{3, 5\};\ 0.05 \leq \alpha_{n'} \leq 0.15 \\ &n'' = \{1, 3, 5\};\ \alpha_{n''} = 1 | n'' = 1;\ 0.005 \leq \alpha_{n''} \leq 0.15 | n'' = \{3, 5\} \end{aligned} \tag{3.6}$$

Finalmente, los parámetros comunes para las perturbaciones relacionadas con notches se representan en (3.5) con $w_n = 2\pi f_n$.

$$\begin{aligned} &0.01T \leq t_d - t_c \leq 0.005T \\ &t_d \leq s;\ t_c \geq 0;\ 0.1 \leq k \leq 0.4; c = \{1, 2, 4, 6\};\ s = \frac{T}{c} \end{aligned} \tag{3.7}$$

Cabe mencionar que los parámetros y características anteriores siguen el estándar IEEE 1159 [67]. La Figura 3.1 muestra de manera gráfica los eventos PQ de la Tabla 3.1.

3.3. PQ-SyDa: Power Quality Synthetic Disturbances DataSet

Una vez definidos los modelos numéricos, se desarrolla una interfaz gráfica de usuario (GUI) basada en Python (el código fuente y la guía de usuario/instalación están

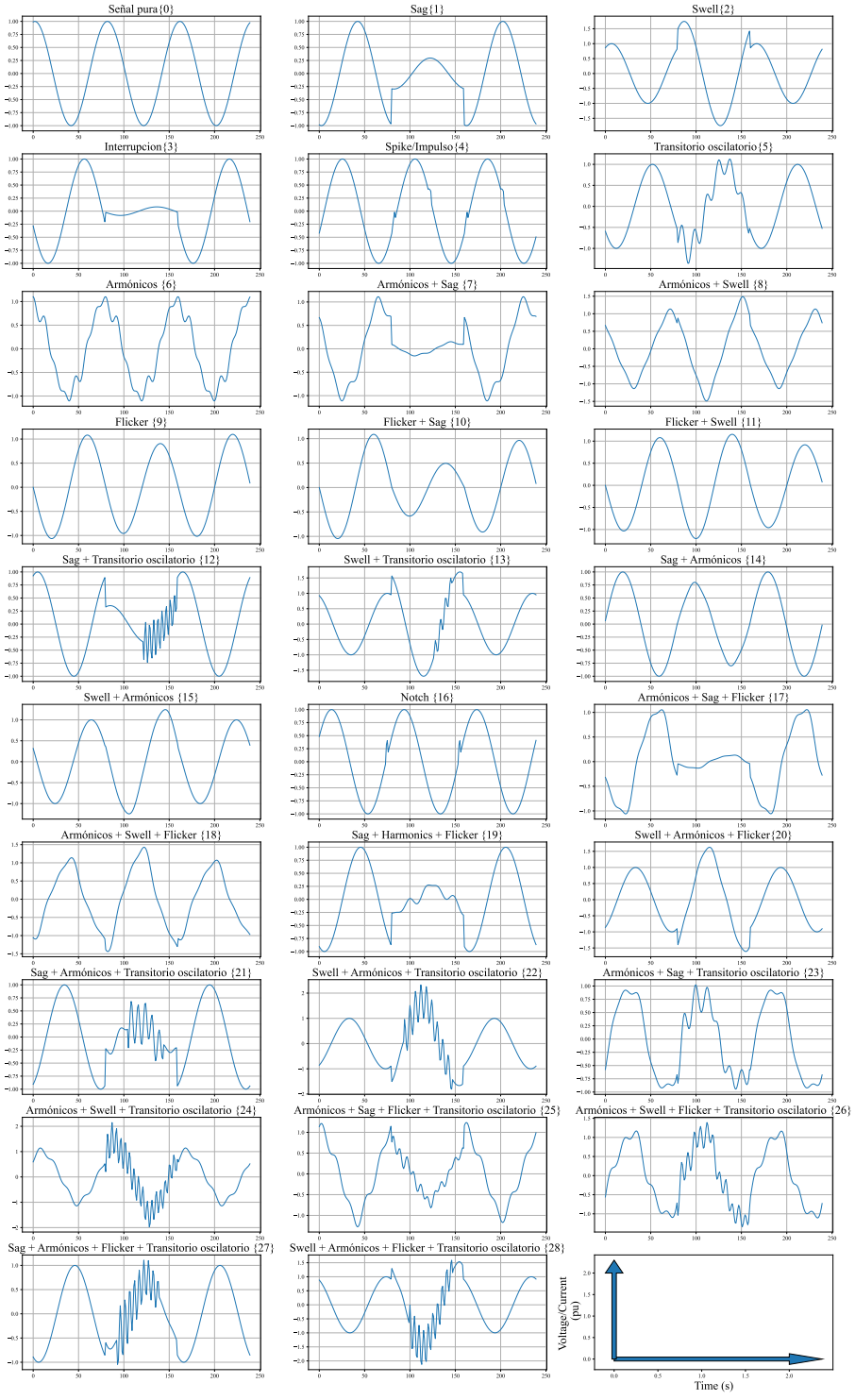


Figura 3.1: Señales de eventos PQ.

disponibles en [64]). La GUI implementada permite la especificación de los parámetros relevantes para construir el conjunto de datos sintéticos.

3.3.1. Interfaz gráfica de usuario

Esta interfaz gráfica de usuario está desarrollada en Python 3.11.5 utilizando principalmente las siguientes librerías:

1. Customtkinter/Tkinter: Para proporcionar widgets de aspecto moderno y totalmente personalizables para la interfaz gráfica.

2. NumPy: Para definir las herramientas y algoritmos matemáticos.

3. Matplolib: Para crear y trazar los gráficos PQ.

4. Pandas/SciPy: Para proporcionar manipulación y portabilidad de conjuntos de datos.

De manera general, el toolbox propuesto se compone principalmente de dos clases: (*i*) una ventana principal y (*ii*) una ventana de exportación, las cuales se describen con detalle a continuación.

3.3.2. Ventana principal

Esta ventana contiene dos secciones principales. La primera establece la configuración de la simulación y los parámetros del modelo, mientras que la segunda comprende la visualización de los eventos generados. Ambas secciones se muestran en la parte superior e inferior de la Fig. 3.2.

Configuración de la simulación y parámetros del modelo

La configuración de la simulación se encuentra en la parte superior de la ventana principal. Así, de izquierda a derecha, se encuentran los siguientes elementos:

Figura 3.2: Ventana principal de la GUI.

- *Duración del evento (ciclos)*. Esta entrada manual recibe un tipo de dato float o int y permite al usuario elegir la duración de la señal de los eventos PQ generados. La longitud de la ventana de la senãl está en ciclos.

- *Frecuencia fundamental (Hz)*. Esta entrada manual recibe un tipo de dato float o int y permite al usuario elegir la frecuencia fundamental en Hertz para generar los eventos PQ.

- *Frecuencia de muestreo (Hz)*. Esta entrada manual recibe un tipo de dato float o int y facilita al usuario elegir la frecuencia de muestreo en Hertz para generar los eventos PQ.

- *Tiempo de inicio de la perturbación (ciclos)*. Esta entrada manual recibe un tipo de dato float o int y permite elegir el tiempo de inicio de la perturbación, especialmente

para eventos transitorios. El tiempo de inicio de la perturbación debe elegirse dentro de la duración del evento; de lo contrario, aparecerá una ventana de error.

- *Duración de la perturbación (ciclos).* Esta entrada manual recibe un tipo de dato float o int y permite al usuario elegir la duración del tiempo de perturbación, especialmente para eventos transitorios.

- *Aleatorio [relacionado con la perturbación].* Esta entrada de cuadro permite la activacioń de la opción aleatoria relacionada con los tiempos de inicio y duración de la perturbación. Si la entrada está habilitada, los tiempos de inicio y duración de la perturbación serán valores aleatorios; de lo contrario, el usuario puede modificar estos valores.

Una vez que se especifican los ajustes de simulación, el usuario puede crear el modelo haciendo clic en el botón **Inicializar modelo**. Este botón también carga aleatoriamente los parámetros de cada perturbación según (3.1) - (3.7). Si el usuario desea otras perturbaciones, es necesario hacer clic nuevamente en el botón **Inicializar modelo** y aparecerán nuevos parámetros.

Parámetros del modelo

Esta caja de herramientas tiene como objetivo crear conjuntos de datos robustos y realistas de eventos PQ en sistemas de potencia. Por lo tanto, el conjunto de parámetros en los modelos numéricos presentados en la Sección 3.2 debe ser aleatorio bajo ciertos límites como se estipula en (3.1) - (3.7). Sin embargo, para algunos casos específicos, la caja de herramientas propuesta le otorga al usuario cambiar los límites inferior y superior de algunos parámetros, como se ve en las pestañas que se muestran en la Fig. 3.2.

Para cambiar los parámetros, se debe presionar el botón **Inicializar modelo** y luego se debe deshabilitar el cuadro de entrada de parámetros predeterminados. Esto permite al usuario ingresar los nuevos parámetros en las diferentes entradas manuales de

cada pestaña. Las descripciones de las diferentes pestañas se presentan a continuación, donde sus entradas manuales son de tipo de datos float o int.

- *General*. En esta pestaña, el usuario puede establecer los límites inferior y superior de la fase (Fase[Mín][Máx]) que es común para todas las perturbaciones; consulte la Fig. 3.3. Luego, la caja de herramientas generará un conjunto aleatorio de eventos que varían este parámetro en el rango seleccionado.

Figura 3.3: Pestaña General.

- *Armónicos*. En esta pestaña, los límites inferior y superior de la amplitud armónica se pueden modificar ([Mín][Máx] para cada armónico) para obtener amplitudes armónicas aleatorias en ese rango; ver la Fig. 3.4. Observe que solo se consideran los armónicos tercero, quinto y séptimo.

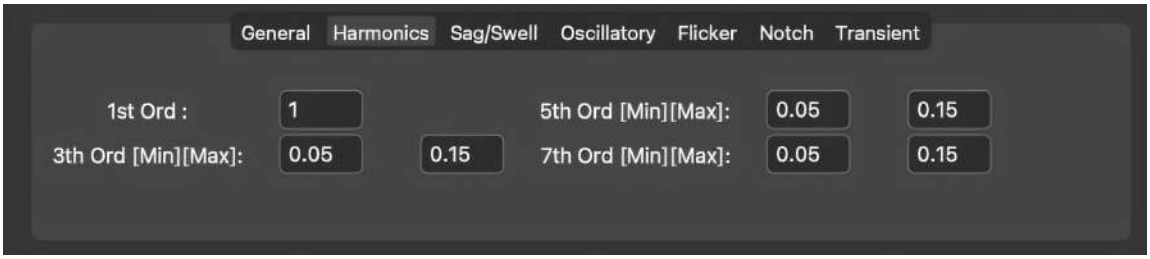


Figura 3.4: Pestaña Armónicos.

- *Sag/Swell*. En esta pestaña, el usuario puede establecer los límites inferior y superior de los parámetros ρ, α y β relacionados con los eventos de sag y swell; ver (3.2) y la Fig. 3.5.

Figura 3.5: Pestaña Sag/Swell.

- *Oscilatorio*. En esta pestaña, el usuario puede establecer los límites inferior y superior de los parámetros relacionados con los eventos transitorios oscilatorios; consulte la Fig. 3.6.

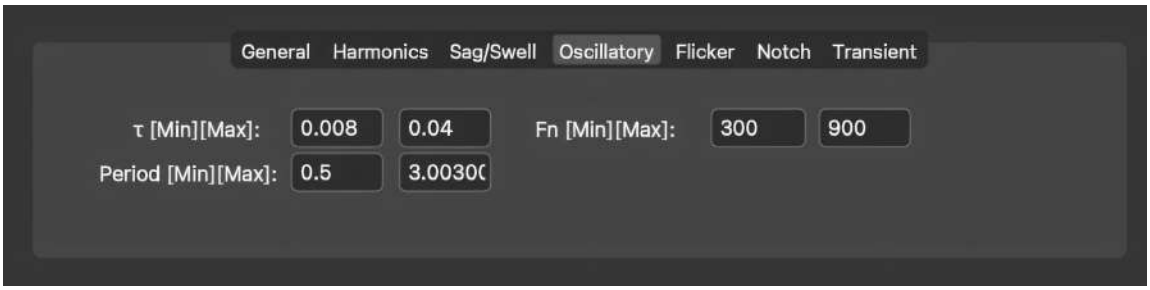


Figura 3.6: Pestaña Oscilatorio.

- *Flicker*. En esta pestaña, el usuario puede establecer los límites inferior y superior de los parámetros relacionados con los eventos de parpadeo, como la frecuencia de flicker; consultar la Fig. 3.7.

- *Notch*. En esta pestaña, el usuario puede establecer los límites inferior y superior del parámetro K relacionado con los eventos de notch; ver la figura 3.8.

- *Transitorio*. En esta pestaña, el usuario puede establecer los límites inferior y superior de los parámetros relacionados con los eventos transitorios; consultar la figura 3.9.

Una vez que el usuario ingresa los límites inferior y superior para cada pestaña, se

Figura 3.7: Pestaña Parpadeo

Figura 3.8: Notch tab.

Figura 3.9: Pestaña transitorios.

debe hacer clic en el botón de cambio de parámetros para aplicar estos cambios, y nuevamente, en el botón **Inicializar modelo** para crear los nuevos modelos. Ahora, el usuario establecerá las configuraciones. El siguiente paso es hacer clic en el botón **EJECUTAR** para generar los 29 eventos aleatorios de acuerdo con la Tabla 3.1. Estos se pueden visualizar individualmente en la sección de **visualización de señales** (ver Fig. 3.2), eligiendo el evento de interés en el botón de menú **Evento PQ**. Si el usuario desea visualizar otra señal

entre los 29 eventos del menú de eventos PQ, solo es necesario elegir el evento y hacer clic en el botón **REFRESH**. La señal aparecerá en la sección de **visualización de señales**.

También, la caja de herramientas propuesta incluye una sección de análisis FFT, donde se puede visualizar el espectro de Fourier, ver Fig. 3.10.

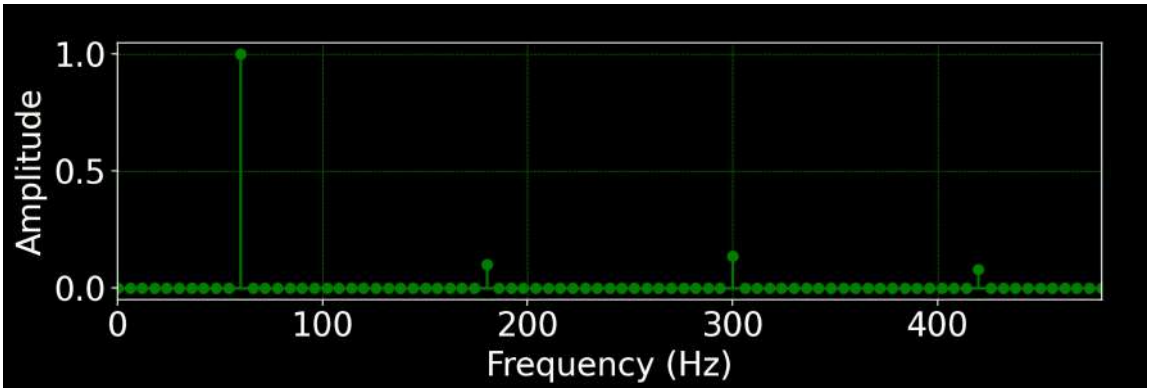


Figura 3.10: Sección FFT para obtener el espectro de Fourier de un evento PQ.

3.3.3. Ventana de exportación

La segunda clase principal de la caja de herramientas propuesta es la función de exportación. Esta se utiliza para exportar el conjunto de eventos deseado por el usuario haciendo clic en el botón **EXPORTAR**. La Fig. 3.11 ilustra el contenido de la ventana de exportación donde, en el lado izquierdo, el usuario puede seleccionar qué eventos desea exportar. En el lado derecho, el usuario puede seleccionar o deseleccionar todos los eventos haciendo clic en los botones **Seleccionar todo** o **Deseleccionar todo**, respectivamente. A continuación, el campo de texto **Dividir porcentaje** permite al usuario definir el porcentaje que se guardará como conjunto de prueba. Para ello, el usuario ingresa un número entre 0 y 1 correspondiente al porcentaje del conjunto de datos de prueba; el resto corresponde automáticamente al conjunto de datos de entrenamiento. Por ejemplo, si el usuario ingresa 0.3, la caja de herramientas genera tres archivos: el primero con el 30 % del conjunto de datos para prueba, el segundo con el 70 % del conjunto de datos para entrenamiento y

el tercero con el conjunto de datos completo. Es importante mencionar que esta selección de porcentaje del conjunto de datos es aleatoria para los primeros dos archivos generados.

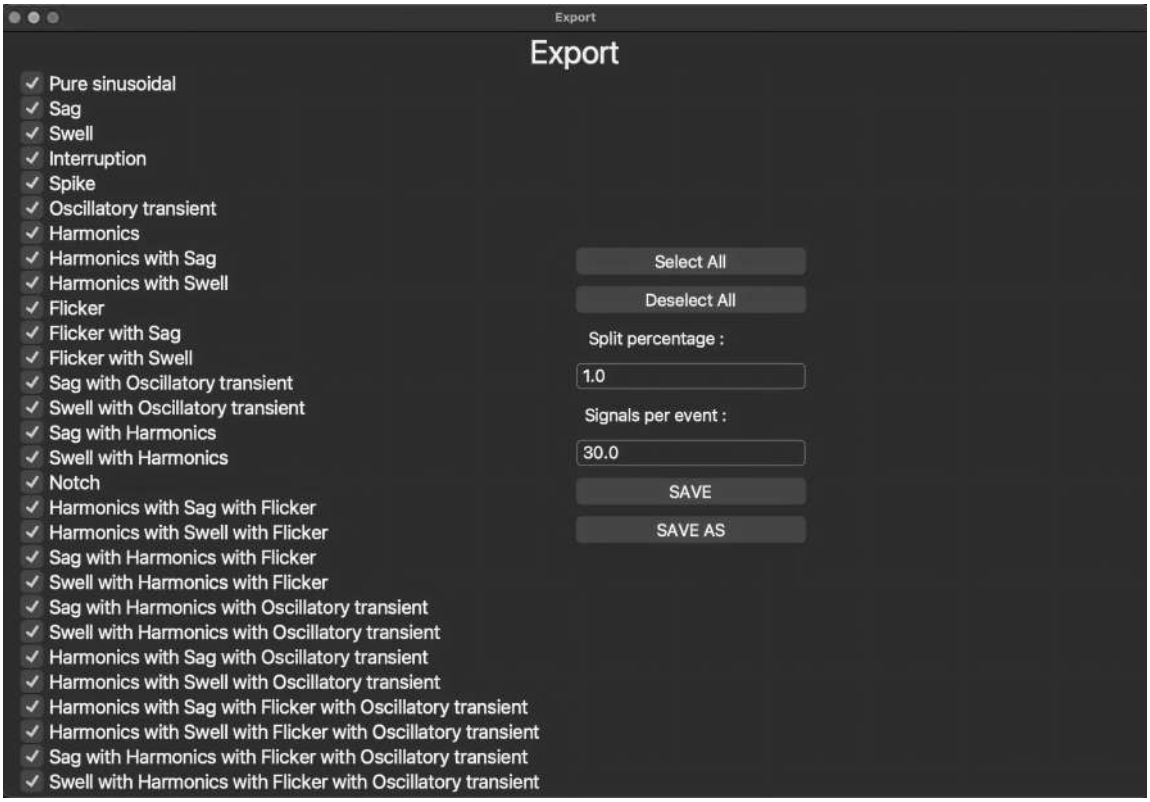


Figura 3.11: Ventana de exportación.

El botón manual de **Señales por evento** permite al usuario definir la cantidad de señales por evento que se incluirán en el conjunto de datos. De esta manera, los elementos en los splits de prueba y entrenamiento se seleccionan aleatoriamente. Finalmente, los botones **GUARDAR** y **GUARDAR COMO**, permiten al usuario guardar el conjunto de datos en una matriz numpy. Además, el toolbox permite utilizar cinco formatos posibles para su exportación y son: *.npy (matriz numpy), *.csv (valores separados por comas), *.txt (archivo de texto), *.mat (archivo Matlab) y *.npz (matriz numpy comprimida).

3.4. Ejemplo de aplicación del Toolbox

Los resultados de un conjunto de datos de muestra para tres modelos de ML se muestran en la pestaña **Aprendizaje automático**. Estos modelos son (*i*) Red neuronal convolucional (CNN), (*ii*) Máquina de vectores de soporte (SVM) y (*iii*) Árbol de decisiones. Los modelos se entrenaron y validaron con un conjunto de datos de eventos PQ sintéticos del Toolbox. Una vez que se completan la configuración y las entradas, solo se eligen las señales para los eventos con las etiquetas 0, 1, 2, 3, 4, 5, 6, 9 y 16 que se muestran en la Tabla 3.1 en la sección de exportación. A continuación, se seleccionan 50 señales en el botón manual **Señal por evento** para obtener 450 señales en el conjunto de datos. Para los conjuntos de datos de prueba y entrenamiento, se selecciona el 20 % ingresando 0.2 en el botón manual **Dividir porcentaje**, mientras que el 80 % restante se genera automáticamente para el conjunto de datos de entrenamiento.

La Figura 3.12 ilustra la ventana que se muestra después de hacer clic en el botón **Modelos** que se muestra en la Figura 3.13, y que se encuentra en la pestaña **Aprendizaje automático** de la ventana principal. De este modo, es posible elegir un modelo entrenado previamente (observe que las etiquetas de dimensión y clase en el conjunto de datos y el modelo deben coincidir). Además, los formatos de archivo admitidos para los modelos de aprendizaje automático son *.pkl, mientras que para el aprendizaje profundo son *.h5.

La predicción generada por el modelo entrenado previamente se muestra en la Figura 3.13. El usuario puede predecir la señal elegida del menú de eventos PQ haciendo clic en el botón **Predecir**. Esto ayuda al usuario a validar cualitativamente el modelo.

Los hiperparámetros utilizados para los modelos que se muestran en la Fig. 3.14 son para el modelo SVM con un núcleo lineal y $C = 1.0$, para el árbol de decisiones, se utilizan el criterio “gini” y el divisor “best”. Para CNN, se utilizan 2 capas de CNN con 32 y 64 filtros cada una y una agrupación máxima en el medio con dos capas densas con 9 y 50 neuronas. Además, se toman para el entrenamiento una tasa de aprendizaje de 1×10^{-6} para 150 épocas y un tamaño de lote de 100 señales. La Figura 3.14 representa

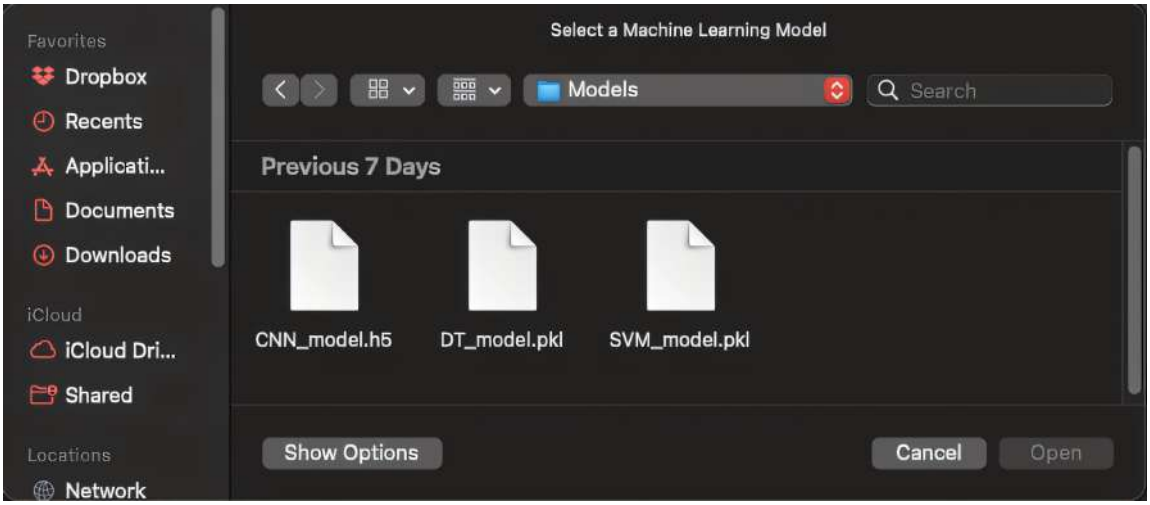


Figura 3.12: Cargar modelos ML y DP

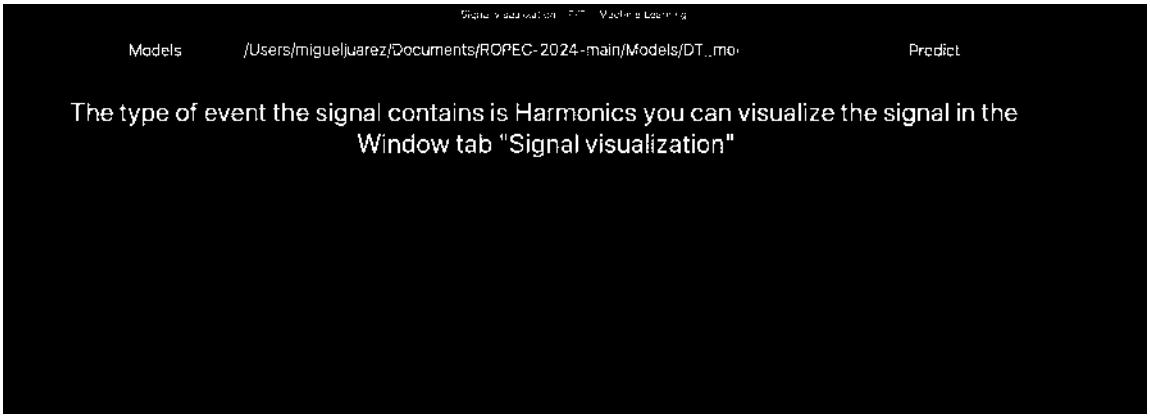


Figura 3.13: Predecir con un modelo ML o DP.

la precisión de cada modelo medida con la puntuación **f1** para cada evento en el conjunto de datos. El ejemplo muestra cómo utilizar el conjunto de datos de eventos de calidad de energía sintética generado por la caja de herramientas para entrenar y probar modelos de aprendizaje automático y profundo.

La falta de bases de datos estándar que puedan utilizarse como puntos de referencia impide la comparación directa de diferentes enfoques. Esta herramienta está destinada

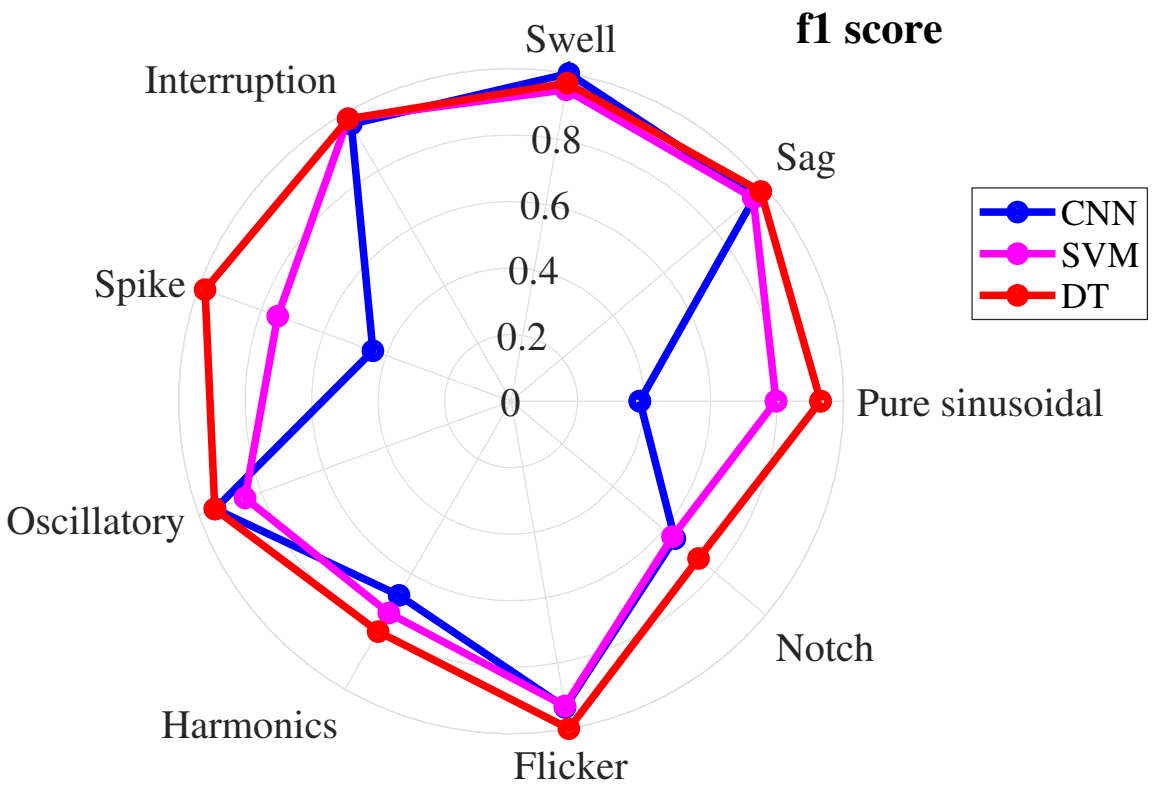


Figura 3.14: Comparación de los modelos pre-entrenados de aprendizaje automático.

a ayudar al desarrollo y prueba de algoritmos basados en datos para detectar y clasificar eventos de calidad de la energía. Su objetivo es proporcionar un punto de comparación para múltiples algoritmos. Permite a los investigadores en este campo generar conjuntos de datos de entrenamiento y validación para evaluar sus algoritmos y compararlos con otros métodos.

Además, la interfaz permite al usuario modificar varios parámetros relacionados con todas las señales y parámetros relacionados con perturbaciones comunes en algunas senãles, algo que puede ser particularmente útil para cada investigador. Por ejemplo, se pueden generar conjuntos de datos sintéticos para que coincidan con senãles reales, que se pueden utilizar conjuntamente para lograr el aumento de datos para modelos profundos.

Así, con la ayuda de este toolbox, es que se van a generar los conjuntos de datos

para entrenar el modelo propuesto para la identificación y clasificación de eventos de calidad de energía presentes en SEPs.

Capítulo 4

Aprendizaje por transferencia y sintonización fina para eventos en la calidad de energía

En este Capítulo se presenta la creación y entrenamiento del modelo de red neuronal convolucional propuesto, el cual se basa en el aprendizaje por transferencia y sintonización fina para la identificación y clasificación de eventos de calidad de energía.

Para la creación del modelo, la identificación y clasificación de eventos PQ, esta tesis adopta la frase de "Divide y vencerás", la cual consiste en entrenar submodelos para posteriormente adoptar la frase "La unión hace la fuerza", que consiste en crear un modelo más grande a partir de los submodelos previamente entrenados. Lo anterior se realiza mediante la combinación de los conceptos contenidos en los Capítulos 2 y 3.

4.1. Divide y vencerás

Para esta etapa de la creación del modelo, se utilizan 29 eventos de calidad de la energía que se pueden presentar en los sistemas de potencia, las cuales son generadas por

medio de PQ-SyDa presentado en el Capítulo 3. Así, estas señales de prueba se pueden catalogar como sigue de acuerdo a [2, 3, 4, 5, 6]:

Eventos que se pueden catalogar como simples:

1. Señal pura.
2. Sag.
3. Swell.
4. Interrupción.
5. Impulso.
6. Transitorio Oscilatorio.
7. Armónicos.
8. Flicker.
9. Notch.

Eventos que se pueden catalogar como compuestos con hasta 2 eventos:

1. Armónicos con Sag.
2. Armónicos con Swell.
3. Flicker con Sag.
4. Flicker con Swell.
5. Sag con Transitorio Oscilatorio.
6. Swell con Transitorio Oscilatorio.

7. Sag con Armónicos.

8. Swell con Armónicos.

Eventos que se pueden catalogar como compuestos con hasta 3 eventos:

1. Armónicos con Sag y Flicker.

2. Armónicos con Swell y Flicker.

3. Sag con Armónicos y Flicker.

4. Swell con Armónicos y Flicker.

5. Sag con Armónicos y Transitorio Oscilatorio.

6. Swell con Armónicos y Transitorio Oscilatorio.

7. Armónicos con Sag y Transitorio Oscilatorio.

8. Armónicos con Swell y Transitorio Oscilatorio.

Eventos que se pueden catalogar como compuestos con hasta 4 eventos:

1. Armónicos con Sag, Flicker y Transitorio Oscilatorio.

2. Armónicos con Swell, Flicker y Transitorio Oscilatorio.

3. Sag con Armónicos, Flicker y Transitorio Oscilatorio.

4. Swell con Armónicos, Flicker y Transitorio Oscilatorio.

Así, el modelo que se propone para la identificación y correcta clasificación de los eventos de calidad de la energía como se menciona en el Capítulo 1, consiste en dividir las señales de prueba de acuerdo al número de eventos que estas señales contienen, con 4 eventos como el número máximo de eventos que se pueden presentar en una señal. A esta

división la llamaremos categoría, es decir, si la señal contiene un evento, corresponde a categoría uno. Si tiene dos eventos, es categoría dos, y así sucesivamente.

Una vez divididas la señales en categorías, se crean 4 modelos, los cuales se detallan más adelante. Estos modelos se entrenan de forma independiente como una técnica de preparación para poder transferir lo aprendido a un modelo más grande ("La unión hace la fuerza), el cual se vuelve a entrenar para finalmente aplicar la técnica de sintonización fina "fine tunning".

Los modelos se entrenan con un conjunto de datos por categoría, este conjunto de datos contiene 1,000 señales de muestra y todos son generados por medio de PQ-SyDa. Por ejemplo, para la categoría 1 se tienen 9 señales con 1 evento cada una, así, este primer modelo se entrena con 9,000 señales de muestra, todas las señales con el mismo número de muestras por señal, a la misma frecuencia y por lo tanto, con la misma frecuencia de muestreo. En este Capítulo, se muestran los resultados de un modelo entrenado con señales con una frecuencia de $50Hz$, una frecuencia de muestreo de $4000Hz$ y una ventana de análisis de 3 ciclos, es decir, 80 muestras por ciclo. Se utiliza esta ventana de análisis, debido a que es la que presenta mejores resultados en cuanto a costo computacional y eficiencia en los modelos, como se muestra más adelante en la Tabla 4.2. Además, en el **Apéndice A** se presentan los resultados obtenidos para los modelos entrenados con la misma frecuencia, misma frecuencia de muestreo y con ventanas de análisis de 1, 2, 4, 5, 6, 7, 8, 9 y 10 ciclos. Así, una vez entrenado cada uno de los modelos utilizando los parámetros anteriores, se prueba con 500 muestras por evento para comprobar su eficiencia.

La Tabla 4.1 muestra la lista de señales y la etiqueta con la que se asocia dicha señal, esta etiqueta se respeta para todas las tablas y figuras posteriores. Observe que la Tabla 4.1 contiene la misma información presentada en la Tabla 3.1 del Capítulo 3.

Etiqueta	Tipo de Señal
0	Señal Pura
1	Sag
2	Swell
3	Interrupción
4	Impulso
5	Transitorio Oscilatorio
6	Armónicos
7	Armónicos con Sag
8	Armónicos con Swell
9	Flicker
10	Flicker con Sag
11	Flicker con Swell
12	Sag con Oscilatorio
13	Swell con Oscilatorio
14	Sag con Armónicos
15	Swell con Armónicos
16	Notch
17	Armónicos con Sag y Flicker
18	Armónicos con Swell y Flicker
19	Sag con Flicker y Armónicos
20	Swell con Flicker y Armónicos
21	Sag con Armónicos y Oscilatorio
22	Swell con Armónicos y Oscilatorio
23	Armónicos con Sag y Oscilatorio
24	Armónicos con Swell y Oscilatorio
25	Armónicos con Sag, Flicker y Oscilatorio
26	Armónicos con Swell, Flicker y Oscilatorio
27	Sag con Armónicos, Flicker y Oscilatorio
28	Swell con Armónicos, Flicker y Oscilatorio

Tabla 4.1: Lista de señales y etiqueta para su identificación.

4.1.1. Modelo categoría 1: señales con un evento

La Figura 4.1 muestra el modelo categoría 1 que se utiliza para clasificar eventos simples. Este modelo contiene 64 filtros en la primer capa convolucional, 128 filtros en la

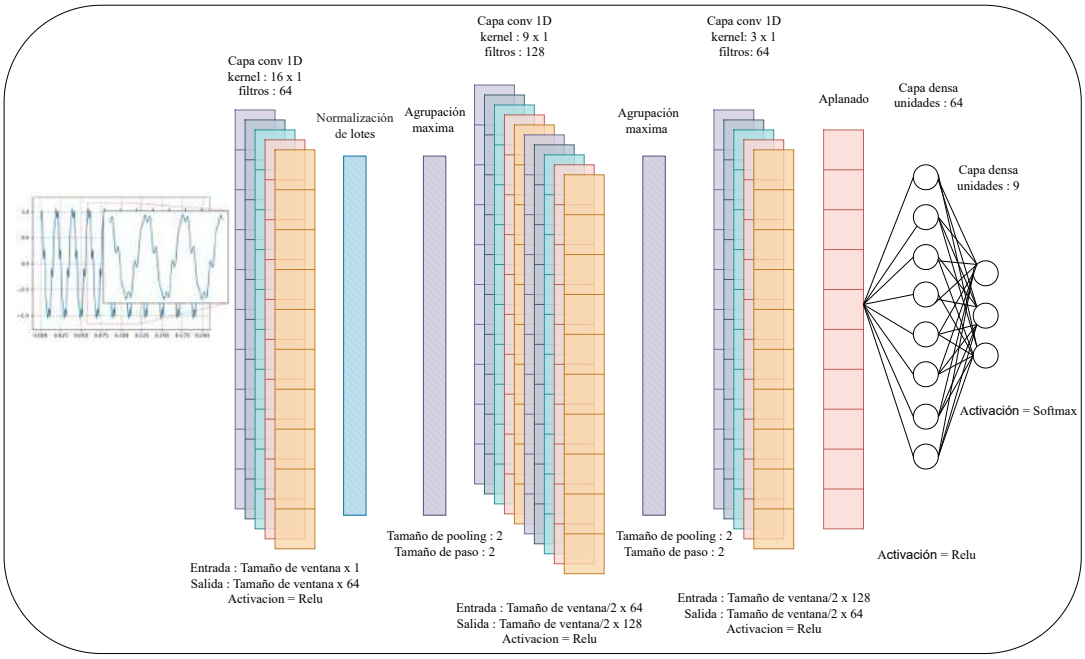


Figura 4.1: Modelo secuencial categoría 1 para clasificar eventos simples.

segunda capa convolucional, 64 filtros en la tercera capa convolucional y 64 unidades en las capas densas. Como función de activación para todas las capas, se utiliza la función Relu mostrada en (2.9). La última capa contiene únicamente 9 unidades con la función de activación softmax, (2.14), ya que es el número de clases de señales que contiene la categoría 1, es decir, 9 eventos simples.

La función de error que se utiliza para este modelo se muestra en (2.17), y el optimizador que se utiliza es el algoritmo de optimización avanzado “Adam” [47], que se encuentra implementado en la librería de acceso libre de TensorFlow [39].

Los resultados para la categoría 1, eventos simples, se detallan a continuación. Con un entrenamiento de 50 épocas, se logra obtener una exactitud del 97.8 % y una pérdida de información del 10 %, una vez entrenado. La Figura 4.2 muestra los resultados obtenidos con un conjunto de datos de 500 señales de muestra por evento en una matriz de confusion,

en el cual las coordenadas en el eje x indican la clase predicha, mientras que el eje y indica la clase a la que pertenece.

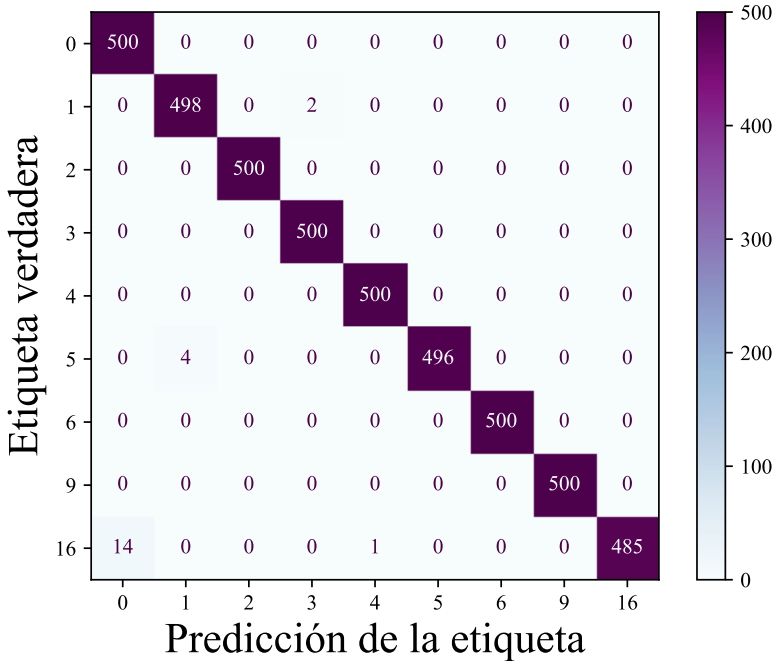


Figura 4.2: Resultados del modelo categoría 1, eventos simples.

Además, los resultados que se obtienen mediante la función $f1$ score usando (2.34), son del 99.46 %. Esto quiere decir, que con los datos de validación se obtiene una exactitud casi del 100 %. Para validar los datos de una mejor manera, se utilizan 10 conjuntos de datos de 500 muestras por señal, todas sintéticas para simular una estimación monte carlo, así, el algoritmo alcanza con esta estimación una media de 99.47 %.

4.1.2. Modelo categoría 2: señales con dos eventos

La Figura 4.3 muestra el modelo que se utiliza para clasificar señales con 2 eventos. Este modelo contiene 64 filtros en la primer capa convoluciuonal, 256 en la segunda capa

convolucional, 64 en una tercera capa convolucional y 64 unidades en la capa densa. Para las capas anteriores, se utiliza la función Relu, (2.9), como función de activación. De acuerdo a la división de eventos, la última capa contiene 8 unidades que es el número de clases que se tienen catalogadas como señales complejas con hasta dos eventos.

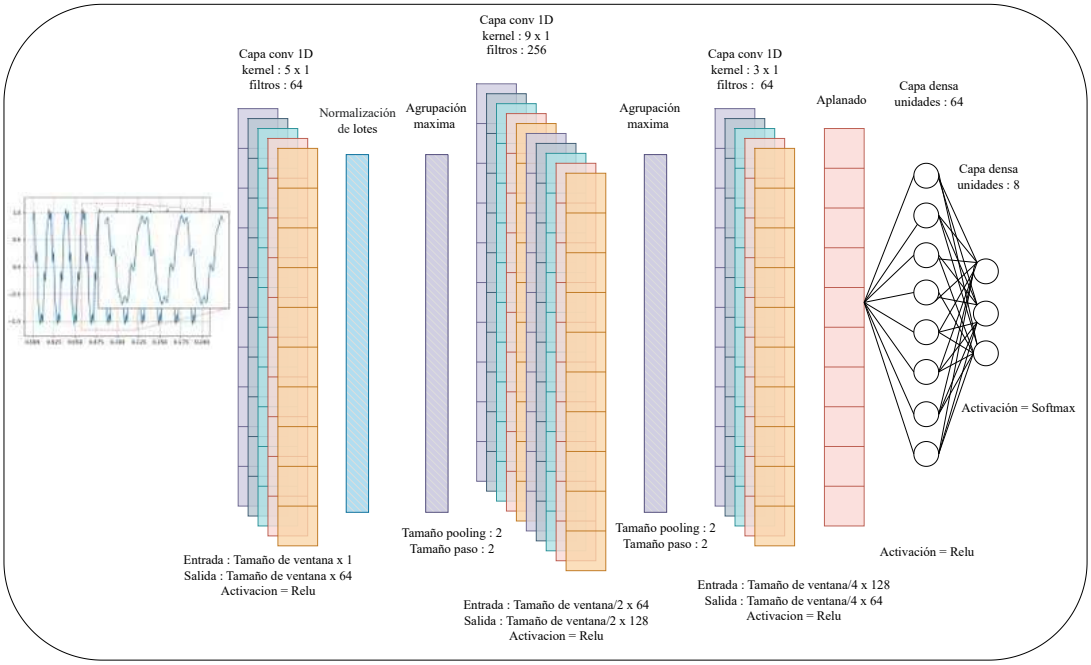


Figura 4.3: Modelo secuencial categoría 2 para clasificar 2 eventos complejos.

La función de error que se utiliza para este modelo, es la misma que la del modelo anterior y se muestra en (2.17), el optimizador que se utiliza para el entrenamiento de este modelo es el algoritmo “Adam” [47]. Así, los resultados para el modelo categoría 2 se detallan a continuación.

Con un entrenamiento de 50 épocas y lotes de 100 señales por iteración en el entrenamiento, se logra una exactitud del 97.28 % y una pérdida de información del 6.57 %, ya una vez entrenado. La Figura 4.4 muestra los resultados obtenidos con un conjunto de datos que contiene 500 señales de muestra por evento, los cuales se representan mediante un

mapa de calor, en la cual las coordenadas en el eje x nos indica la clase predicha, mientras que el eje y, la clase a la que pertenece.

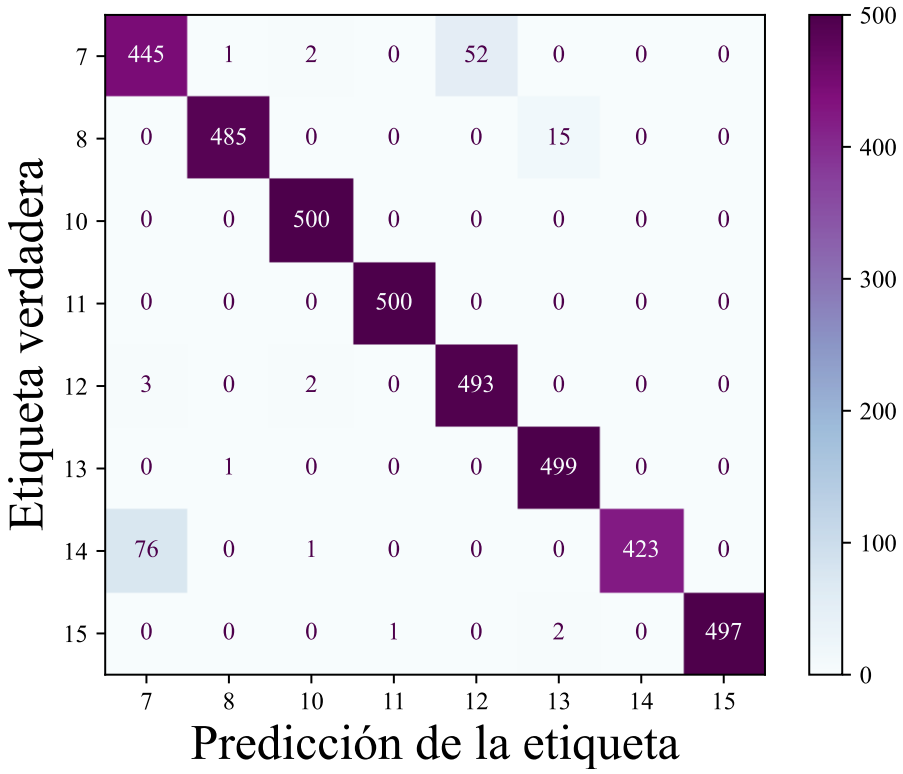


Figura 4.4: Resultado del modelo categoría 2, señales complejas con dos eventos.

Por otro lado, los resultados obtenidos mediante la función $f1$ score son del 96.27 %, lo cual indica, que con los datos de validación, su rendimiento fue muy bueno. Esto es, como se puede observar en la Figura 4.4 y lo reportado por la función $f1$ score, la eficiencia del algoritmo es muy buena, los resultados obtenidos son considerablemente buenos. Cabe mencionar que se realizó la prueba con 10 conjuntos de prueba de 500 señales de muestra por evento, y en las 10 ocasiones el resultado promedio fue del 96.7 %.

Además, algo interesante que se presenta en específico para este caso de estudio,

es de que las clases etiquetadas como {7} y {14} varían únicamente en el momento en el que inician los eventos, es decir, la clase {7} es *Armónicos con Sag*, mientras que la clase {14}, es un *Sag con Armónicos*, por lo que la red se equivocó en 47 y 74 ocasiones, respectivamente, para este caso de estudio. Lo anterior es interesante, ya que la extracción de características que hace la red neuronal tiene que ver con la magnitud y frecuencia del evento, tal como lo hace la transformada de Fourier, pero también tiene que ver con el tiempo, tal y como lo hace la transformada Wavelet. En otras palabras, es importante mencionar, que en sí, no es necesaria una extracción de caracteríticas como una etapa de pre-procesamiento, tal y como se hace en [31, 32], ya que la red neuronal lo hace internamente, lo cual es una de las ventajas de los modelos propuestos.

4.1.3. Modelo categoría 3: señales con tres eventos

La Figura 4.5 muestra el modelo que se utiliza para clasificar 3 eventos complejos. El modelo contiene 128 filtros en la primer capa convolucional, 64 en la segunda capa convolucional y 64 unidades en la capa densa se utiliza este modelo ya que tras varios intentos de prueba y error este modelo fue el que dio mejores resultados al hacer la extracción de características. Todas las capas con la función Relu como función de activación. La última capa contiene 8 unidades que corresponde al número de clases de los eventos catalogados como complejos con 3 eventos, para esta capa, se utiliza la función softmax como función de activación.

Los resultados obtenidos para este modelo y utilizando un entrenamiento de 50 épocas y lotes de 100 señales por iteración en el entrenamiento, logra una exactitud del 97.3 % y una pérdida de información del 6.13 %, una vez entrenado. La Figura 4.6 muestra los resultados obtenidos en un mapa de calor, en el cual las coordenadas en el eje x nos indica la clase predicha, mientras que el eje y, la clase a la que pertenece.

Ahora, los resultados obtenidos para el conjunto de validación con la función $f1$ score, son del 98.97 %, es decir, que con los datos de validación se equivoca el modelo en

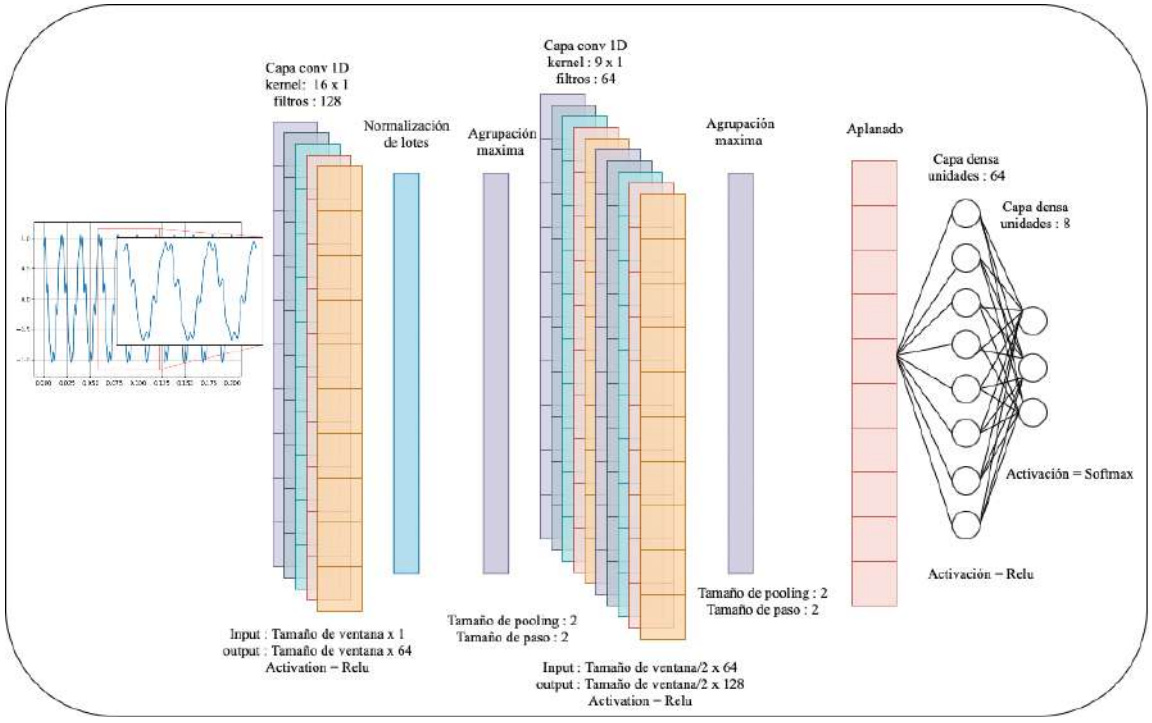


Figura 4.5: Modelo secuencial categoría 3 para clasificar 3 eventos complejos.

promedio, en 8 señales por cada 500, lo cual resulta ser muy bueno. Así, como se puede observar en la Figura 4.6 y lo reportado por la función $f1$ score, la eficiencia del algoritmo es sobresaliente, por lo que, los resultados obtenidos son considerablemente buenos. Cabe mencionar, que se realizó la prueba con 10 conjuntos de prueba de 500 señales muestra por evento, teniendo en promedio una eficiencia del 97.99 %.

4.1.4. Modelo categoría 4: señales con cuatro eventos

Finalmente, la Figura 4.7 muestra el modelo que se utiliza para clasificar señales con 4 eventos complejos. Este modelo contiene 128 filtros en la primer capa convolucional, 64 en la segunda capa convolucional y 64 unidades en las capas densas. Todas las capas anteriors con la función Relu como función de activación. La última capa contiene 4 unidades que es el número de clases que se tienen para esta categoría. Además, la función de

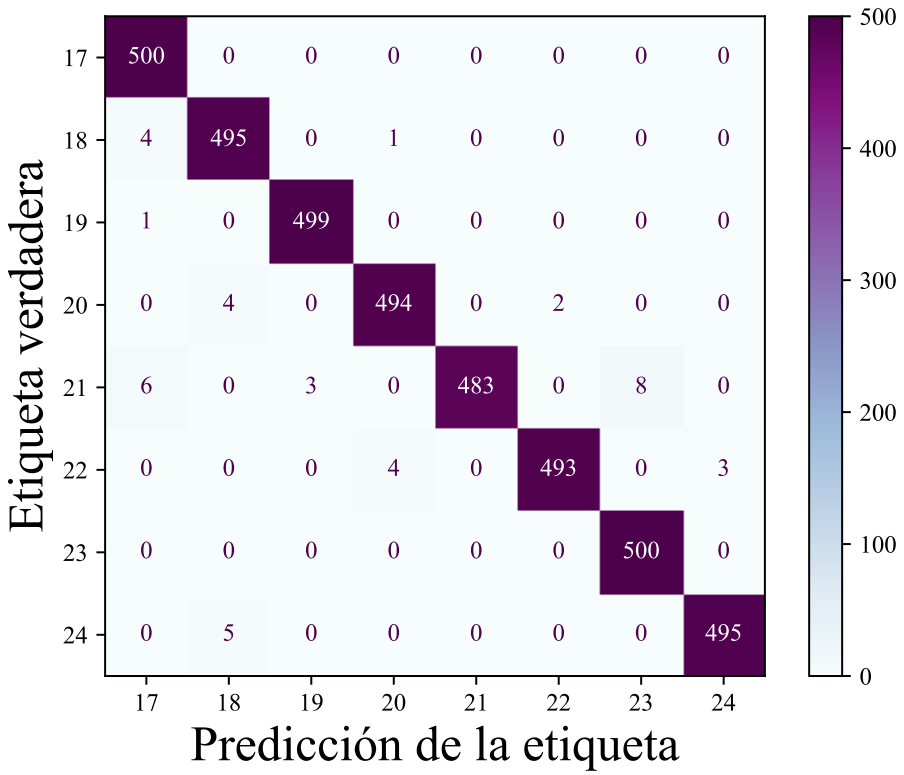


Figura 4.6: Resultados del modelo categoría 3, señales complejas con tres eventos.

error que se utiliza para este modelo se muestra en (2.17), así como, el optimizador que se utiliza es el algoritmo de optimización "Adam".

Así, los resultados para este modelo utilizando un entrenamiento de 50 épocas en lotes de 100 señales, logra una exactitud del 95 % y una pérdida de información del 10 %, una vez entrenado. La Figura 4.8 muestra los resultados obtenidos en un mapa de calor, en el cual las coordenadas en el eje x nos indica la clase predicha, mientras que el eje y, la clase a la que pertenece.

Los resultados obtenidos con la función $f1$ score son del 98.35 %, es decir, que con

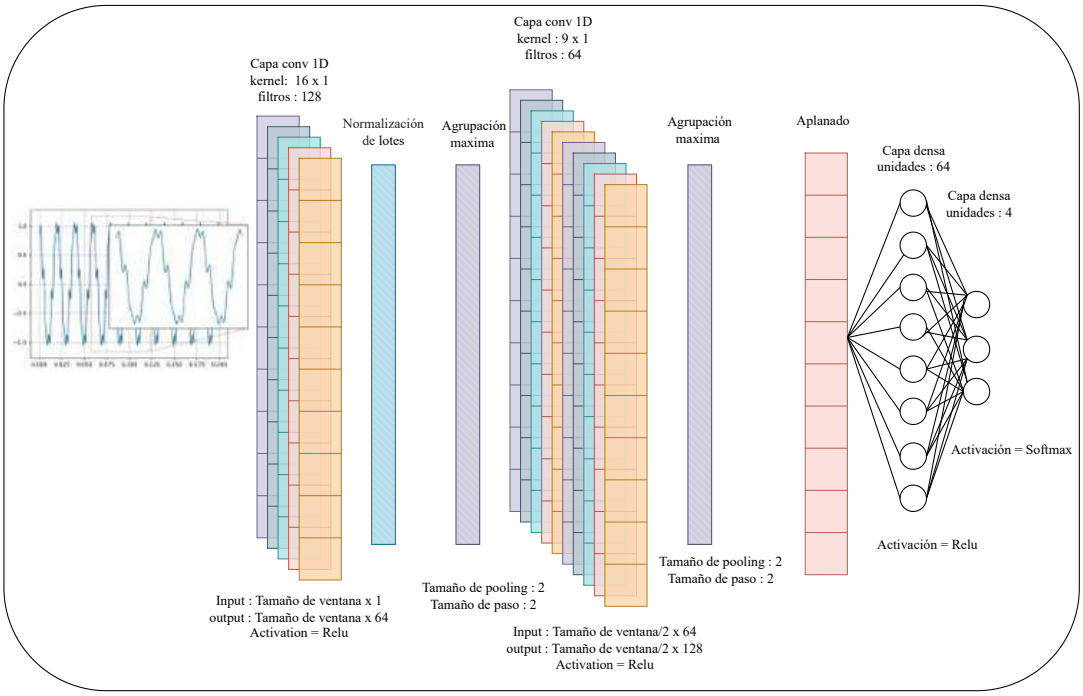


Figura 4.7: Modelo secuencial categoría 4 para clasificar 4 eventos complejos.

los datos de validación los resultados son muy buenos. En la Figura 4.8, se puede observar que debido a lo reportado por la función $f1$ score, la eficiencia del algoritmo es muy buena. Al igual que con los modelos anteriores, se realizón la prueba con 10 conjuntos de prueba de 500 señales de muestra por evento y con 10 conjuntos de datos diferentes, teniendo como resultado un promedio de 98.35 %.

4.2. La unión hace la fuerza

Una vez que se tiene el pre-entrenamiento de los 4 modelos correspondientes para las categorías 1, 2, 3 y 4, respectivamente, se procede a la creación y entrenamiento del modelo propuesto basado en el aprendizaje por transferencia y sintonización fina para la detección y correcta clasificación de las 29 señales sintéticas presentadas en el Capítulo 3.

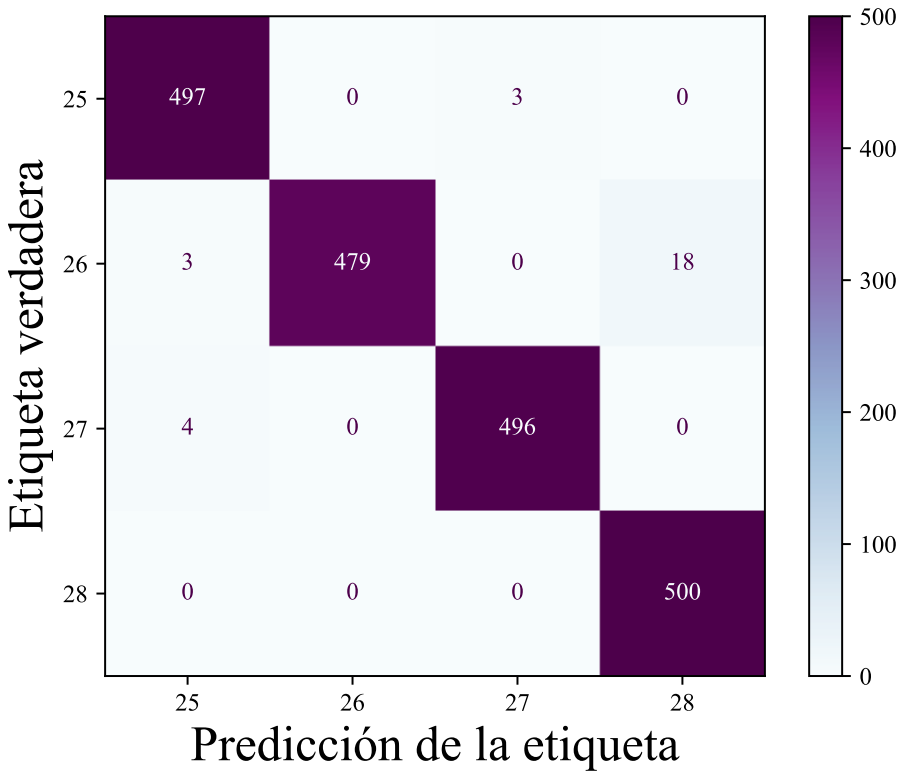


Figura 4.8: Resultados del modelo categoría 4, señales con cuatro eventos complejos.

Así, para la etapa de aprendizaje por transferencia durante el entrenamiento para los modelos de las diferentes categorías, se utiliza un conjunto de datos con 1000 señales de muestra por evento, es decir, un total de 29,000 señales. Para la etapa de validación, se utiliza un conjunto de datos de 500 señales de muestra por evento, es decir, un total de 14,500 señales.

Por otro lado, para la sintonización fina en la etapa de entrenamiento, se utiliza un conjunto de datos del mismo tamaño al de la etapa de transferencia de aprendizaje, es decir, 1000 señales de muestra por evento, mientras que para la etapa de validación, se utiliza un conjunto de datos, también de 500 señales de muestra por evento para llegar a

un total de 14,500 señales.

4.2.1. Creación del modelo propuesto

La Figura 4.9 presenta la unión de los cuatro modelos previamente entrenados, donde se observa que tienen una capa de entrada conectada a cada uno de los modelos y se concatena la salida de los cuatro modelos a este nuevo modelo. De esta manera, las características previamente aprendidas por los filtros se transfieren y adieheren a este nuevo modelo. Como se observa en la Figuras 4.5 y 4.7, los modelos para clasificar señales con 3 y 4 eventos son más pequeños en comparación con el modelo para clasificar señales con 2 eventos mostrado en la Figura 4.3, lo cual se puede decir que es contrario, ya que las características que se pueden presentar en una señal con 3 o 4 eventos es mayor que las características que se pueden presentar en una señal con 2 eventos. Así, los modelos que se muestran en las Figuras 4.5 y 4.7, no mejoran su eficiencia al agregar más capas, aún si se aumenta el número de filtros en las capas que ya tiene, por lo que en conclusión, la extracción de características obtenida por los modelos es suficiente para clasificar las señales con el mismo número de eventos. El detalle, es de que no son suficientes para clasificar señales con un número distinto de eventos, por lo que a este modelo final es necesario agregar dos capas convolucionales con 32 y 128 filtros, respectivamente, con capas de regularización, específicamente, dos capas de agrupación máxima y normalización más una capa de abandono. También se agrega una capa de aplanado, la cual cambia toda la salida a un vector de una dimensión, y se agregan dos capas densas. La última capa con la función softmax como función de activación, mientras que todas las capas mencionadas anteriormente, tienen como función Relu como función de activación.

La Figura 4.9 muestra la estructura final del modelo de la red neuronal convolucional que se utiliza para la clasificación de las 29 señales con diferentes eventos de calidad de la energía. A continuación se detalla la forma en que se utiliza el aprendizaje por transferencia y sintonización fina para el modelo propuesto.

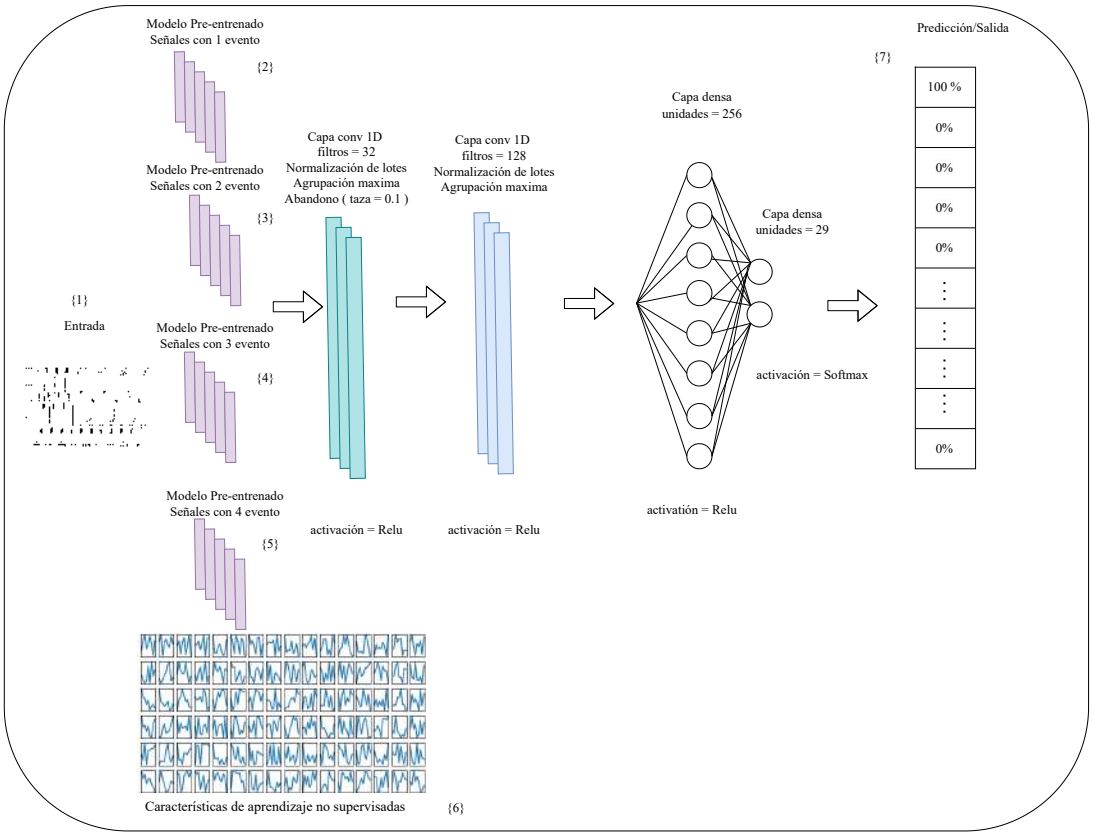


Figura 4.9: Modelo final propuesto para la identificación y clasificación de eventos PQ.

4.2.2. Etapa de aprendizaje por transferencia

El entrenamiento del modelo durante esta etapa es muy similar a los entrenamientos de las redes neuronales anteriores, la diferencia recae en congelar las capas de los modelos previamente entrenados para que no olviden las características que previamente aprendieron, es decir, las características que el modelo previamente aprendió sin supervisión humana. Estas capas no se van a modificar durante el aprendizaje por transferencia. En la Figura 4.9 se aprecian las partes que componen al modelo final propuesto, donde las partes {2},{3},{4},{5} representan las capas que no se van a modificar durante esta etapa

de aprendizaje por transferencia. La parte {6} muestra algunos de los filtros contenidos en las partes {2} a {5}, los cuales ya tienen características importantes que se van a conservar durante esta etapa. Por último, la parte {7} muestra la salida de la red neuronal, que como se observa, es un vector de probabilidades. En algunos casos, especificámente para la clasificación de las clases 7 y 12, este vector tenía probabilidades cercanas a 50 %, por lo que, para automatizar el algoritmo, se toma la de mayor probabilidad.

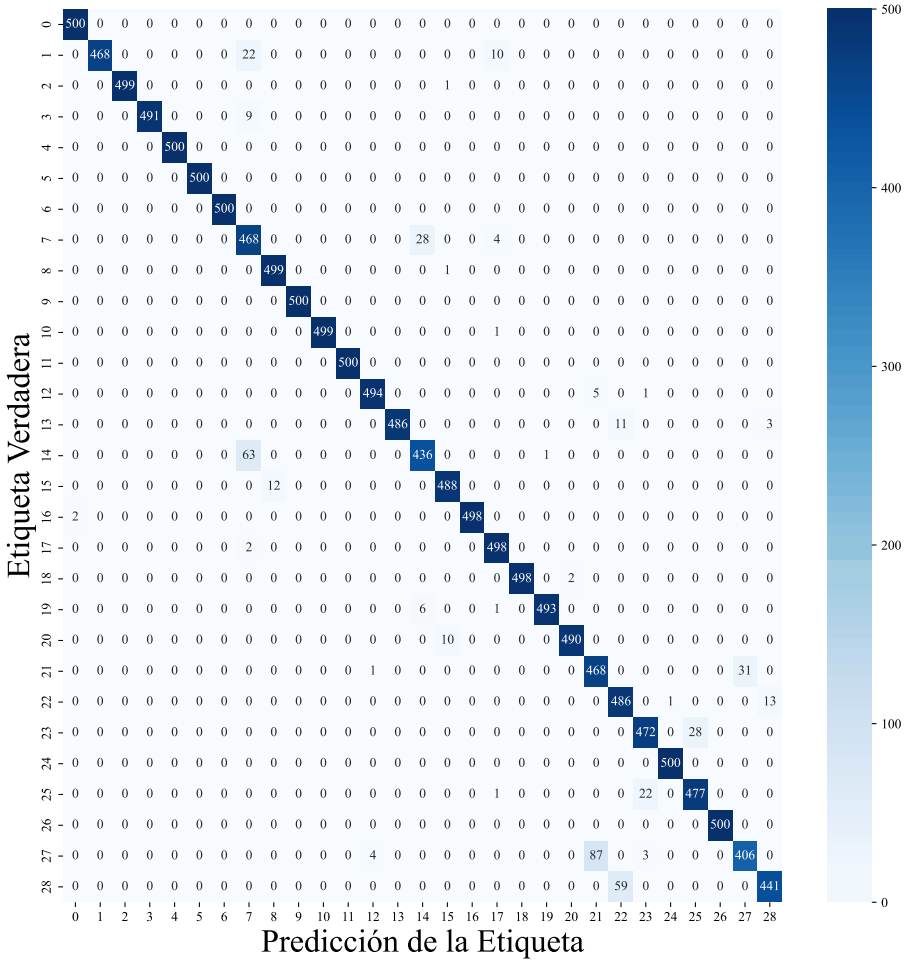


Figura 4.10: Resultados del modelo propuesto después del aprendizaje por transferencia.

La Figura 4.10 muestra los resultados del modelo propuesto después del aprendizaje por transferencia. Como se puede observar, es muy bueno, ya que la red neuronal se equivoca muy poco siendo uno de los principales errores la identificación y correcta clasificación de las clases 7 y 12. Lo anterior, debido a que el modelo las toma como la misma clase. Esto, ya que como se menciona en la Tabla 4.1, la clase 7 es una señal con eventos *Sag con Armónicos* y la clase 12 es una señal con los mismos eventos, pero en orden contrario. Aun así, la eficiencia de la red es muy buena, donde la función $f1$ score presenta una media del 96.39 % de eficiencia en 10 conjuntos de datos con 500 muestras por señal, es decir, 14,500 señales.

4.2.3. Etapa de sintonización fina

Para esta etapa, se descongelan los modelos que se congelaron previamente en la etapa de aprendizaje por transferencia, y se utiliza una tasa de actualización muy baja, con la finalidad de no modificar todas las características que el modelo ha aprendido hasta esta etapa. En este caso, se utiliza 1×10^{-7} como tasa de actualización. y se entrena durante 10 épocas. Así, los resultados del modelo propuesto después de la sintonización fina se muestran en la Figura 4.11.

Aunque la Figura 4.11 en comparación con la Figura 4.10 puede resultar difícil ver la mejora en la eficiencia del modelo propuesto, con la función $f1$ score es más evidente notarla, ya que con 10 conjuntos de datos de 500 señales de muestra por evento, la propuesta alcanza una eficiencia con la métrica f1 score del 97.80 % comparada con el 96.39 % del modelo sin la etapa de sintonización fina, es decir, 1.4 % más eficiente. En este caso específico, con un conjunto de datos de 14,500 señales en total, mejora en la clasificación de 174 señales. Este paso mejora mayormente al modelo para la identificación de las señales con eventos más complejos, es decir, en las señales con 3 o 4 eventos, por lo que si alguien estuviera interesado en clasificar únicamente señales con 1 o 2 eventos, esta etapa no es requerida.

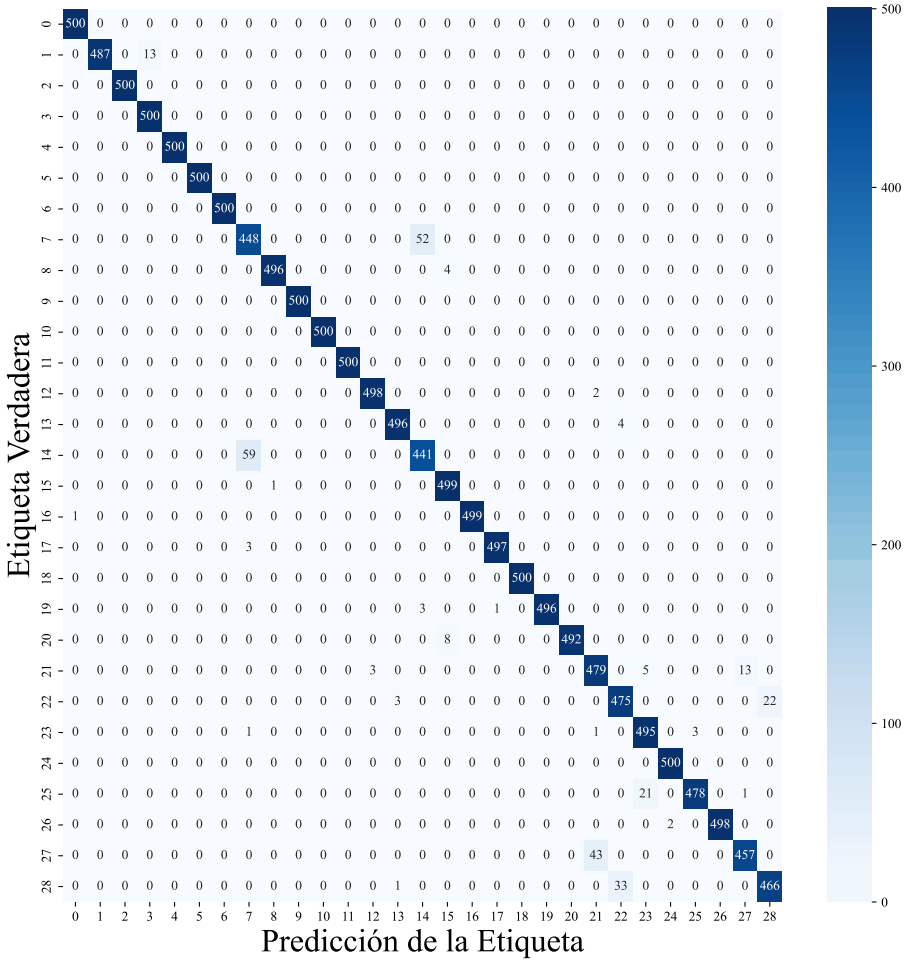


Figura 4.11: Resultados del modelo propuesto después de aplicar la sintonización fina.

4.3. Comparativa con la creación de otros modelos

Como se menciona anteriormente, el modelo propuesto se entrena con una ventana de análisis de 3 ciclos, es decir, 80 muestras por ciclo. Sin embargo, aunque la red neuronal da resultados muy adecuados con 3 ciclos, puede surgir la duda de si con esta cantidad de ciclos es viable una implementación en tiempo real o si es preferible utilizar menos o más ciclos para una mayor eficiencia, o si al reducir o aumentar la cantidad de ciclos disminuye

la eficiencia del modelo propuesto.

Para poder resolver estas preguntas, se debe notar que el modelo crece muy rápido al aumentar la entrada de la red, por ejemplo, la cantidad de parámentros de un modelo para un ciclo es de $660,605$ parámentros, mientras que para un modelo de 10 ciclos, es de $2,135,165$ parámetros, lo que se traduce a un aumento considerable en el costo computacional.

Por lo tanto, esta Sección se centra en realizar una comparación en la eficiencia del modelo final propuesto, ver Figura 4.9, adaptando la red para entradas con 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10 ciclos. La comparativa se centra en 4 características que hacen a un modelo más atractivo que otro al realizar una posible implementación en tiempo real. Además, los resultados de los modelos pre-entrenados para cada una de las entradas con 1 a 10 ciclos, se pueden observar en el Apéndice A.

La Tabla 4.2 muestra la comparativa con las cuatro características importantes: tamaño denotado por la columna párametros, precisión, tiempo para predecir 1 muestra y tiempo para 1000 muestras. Se puede observar que el modelo más eficiente corresponde al modelo de 5 ciclos, aunque los modelos de 4 y 6 ciclos no difieren mucho, esta eficiencia es muy buena. Por otra parte, para el tiempo de ejecución se utiliza un CPU m2 de la marca Mac, con las especificaciones más básicas, el modelo más rápido para predecir 1000 señales es el que tiene una ventana de análisis de 1 ciclo, aunque su eficiencia se reduce casi un $10\,\%$ respecto al modelo de 2 ciclos. Por lo que lo más prudente en cuanto a velocidad al realizar una predicción y una posible implementación en tiempo real, puede ser el modelo para 3 ciclos, el cual es rápido y eficiente.

Cabe mencionar que estos modelos se entrenaron con un conjunto de datos con 1,000 muestras por señal, es decir, se entrenaron con un conjunto de datos de 29,000 señales en la etapa de aprendizaje por transferencia y con 1,000 muestras por señal en la etapa de sintonización fina, todas del tamaño correspondiente al modelo, es decir, una señal de tamaño 80 muestras que es lo correspondiente a un ciclo para el primer modelo, una señal

Modelo	Parámetros	Precisión	Predicción 1 Señal (s)	Predicción 1000 Señales (s)
1 Ciclo	660,605	86.09 %	0.024805	0.218550
2 Ciclos	824,445	95.70 %	0.024715	0.491214
3 Ciclos	**988,285**	**97.80 %**	**0.023799**	**0.545018**
4 Ciclos	1,152,125	98.84 %	0.024204	0.652974
5 Ciclos	**1,315,965**	**98.86 %**	**0.026792**	**0.827320**
6 Ciclos	1,479,805	98.84 %	0.026337	0.913436
7 Ciclos	1,643,645	98.56 %	0.028266	1.007385
8 Ciclos	1,807,485	98.60 %	0.025300	1.134528
9 Ciclos	1,971,325	98.38 %	0.024300	1.236538
10 Ciclos	2,135,165	98.306 %	0.026180	1.528867

Tabla 4.2: Comparativa de la eficiencia de modelos entrenados con 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10 ciclos.

con 120 muestras correspondientes a 2 ciclos para el segundo modelo, y así sucesivamente. Si aumentamos el conjunto de datos para el modelo de 1 ciclo hasta 10,000 muestras por señal, este alcanza una eficiencia del 93 % pero es un modelo no muy adecuado y que no se menciona más a detalle, ya que las señales con 3 o 4 eventos no las puede clasificar bien debido a las limitaciones en la información, además de que esta comparativa no es justa si el tamaño del conjunto de entrenamiento no es igual para todos.

En la Figura 4.12 se muestra una comparativa en cuanto a la eficiencia del modelo, el tamaño del modelo y el tamaño de entrada de cada uno de los modelos. Como se puede observar, el tamaño de la entrada del modelo es linealmente proporcional al tamaño del modelo, es decir, que conforme se aumenta el tamaño de entrada del modelo, el modelo va a crecer linealmente, mientras que la eficiencia no va a aumentar, ya que a partir de 4 ciclos el modelo no mejora pero tampoco empeora de manera significativa. En pocas palabras, no es viable crear modelos más grandes de los que se mencionan en esta Sección. Así, el modelo utilizado en esta tesis para la identificación y clasificación de eventos de calidad de energía es el correspondiente al modelo de 3 ciclos.

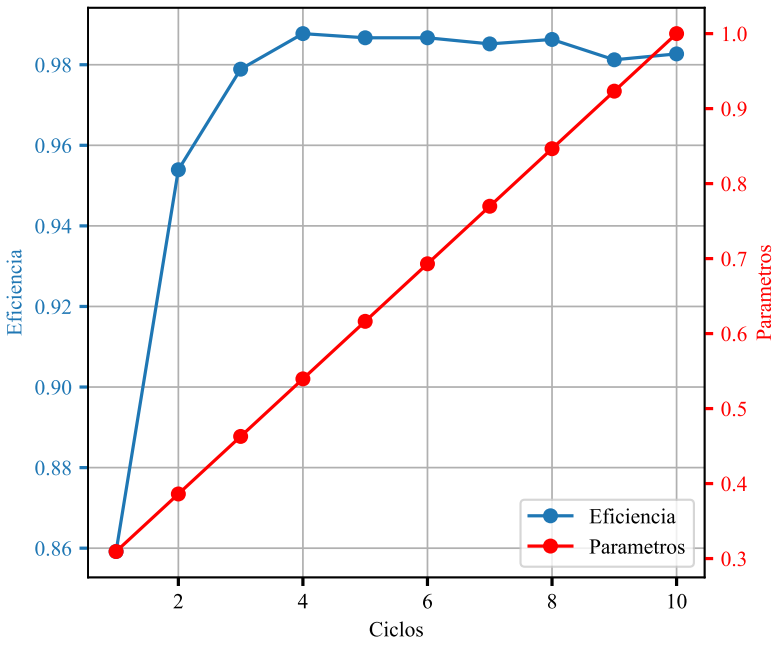


Figura 4.12: Comparativa de los 10 modelos presentados en la Tabla 4.2.

Capítulo 5

Casos de estudio y resultados

Los casos de estudio utilizados para validar la efectividad de la propuesta en esta investigación corresponden a: (*i*) señales sintéticas PQ, (*ii*) eventos PQ obtenidos por sistemas modelados en Simulink y (*iii*) señales reales.

5.1. Resultados con señales sintéticas

Para el caso de las señales sintéticas, los resultados corresponden a los presentados en la Figura 4.11 con una eficiencia del 97.8 % como se muestra en la Tabla 4.2.

Además se hace una breve comparativa con los resultados de algunos algoritmos que actualmente se presentan en el estado del arte, así como una comparativa con otros algoritmos de Machine Learning y Deep Learning que se entrenan con una distribuición de muestras igual a la distribución utilizada con el algoritmo propuesto, esto con la finalidad de tener una comparación justa.

Por otra parte, la Tabla 5.1 muestra una comparativa con los algoritmos que actualmente se han desarrollado para la identificación de eventos PQ, donde la comparativa se realiza con algunos de ellos a pesar de que no se entrenaron ni desarrollaron con las mismas características que este modelo. La finalidad es presentar lo que hay reportado en

la literatura y compararlo con la propuesta (TL&FT-CNN).

Referencia No.	Metodo	Real/Sinteticas/Simuladas	No. de fases	Voltaje/Corriente	No. de PQD	Exactitud
[68]	RN	S	1	V	17	99.66 %
[69]	BCNN	S	1	V	6	99 %
[70]	W-CTN	S	1	V	15	98.67 %
[71]	CNN	S	1	V	29	97.52 %
[72]	VMD-CNN	S	1	V	7	98.7 %
[73]	WT, CNN	S	1	V	7	98.94 %
[74]	DCNN	S	1	V	12	99.61 %
[75]	DCNN	S	1	V	14	99.5 %
[76]	DCNN	S	1	V	28	97.84 %
[77]	CAE, SLSTM	S	1	V	9	98 %
[78]	CNN	S	1	V	6	99.83 %
[79]	EDTCNN	S	1	V	29	99.62 %
[80]	CNN, LSTM	S	1	V	15	99.87 %
[81]	PCNN	S	3	V	19	98 %
[82]	MVNN, CNN	S/R	1	V	5	90 %
[83]	IFCN	S/R	1	V	12	93 %
[84]	CNN	R	1	V/C	21	98.45 %
Este trabajo	TL&FT-CNN	S/S/R	1	V	28	98.86 %

Tabla 5.1: Comparativa con algoritmos del estado del arte.

La Tabla 5.2 muestra una comparativa con 3 algoritmos más de Machine Learning: una red neuronal convolucional, un árbol de decisión (Decision Tree) y una máquina de soporte vectorial (Support Vector Machine), los cuales son entrenados con el mismo número de señales y distribución con la que se entreno el modelo con aprendizaje por transferencia y Fine Tunning (TL&FT-CNN). Las etiquetas mostradas en las siguientes Figuras corresponden a lo mostrado en las Secciones anteriores.

Modelo	Eficiencia (f1 score)
TL&FT-CNN	98.86 %
CNN	95 %
DT	58 %
SVM	48 %

Tabla 5.2: Comparación de la propuesta con algortimos de ML y DL.

El preprocesamiento que se aplica a la máquina de soporte vectorial y al árbol de

decisión es relativamente simple, se extraen las mismas características que se muestran en el Capítulo 3, mientras que la red neuronal convolucional se entrena durante 50 épocas.

La Figura 5.1 muestra una comparativa en gráfico de barras de la eficiencia de los cuatro modelos al predecir cada una de las señales, este resultado es el promedio que obtuvieron al se probados con 500 señales. Como se puede observar, el algoritmo con mayor eficiencia es el algoritmo de red neuronal convolucional con transferencia de aprendizaje y sintonización fina, etiquetada como CNNFT, mientras que el peor fue la máquina de soporte vectorial.

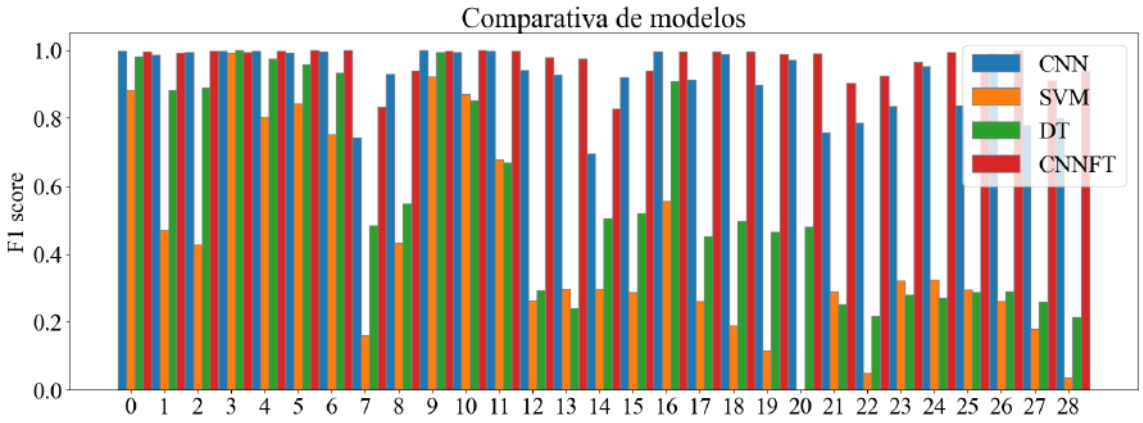


Figura 5.1: Comparativa con otros algoritmos de Machine Learning.

La Figura 5.2 muestra una comparativa en gráfico de telaraña, en esta gráfica es más notorio que la red neuronal convolucional con transferencia de aprendizaje y sintonización fina, etiquetada como CNNFT, tuvo mejores resultados que la red neuronal convolucional normal.

5.2. Resultados con señales simuladas

Para probar si la red neuronal propuesta obtiene buenos resultados en un entorno más real, se prueba el modelo con señales obtenidas por medio de sistemas modelados en

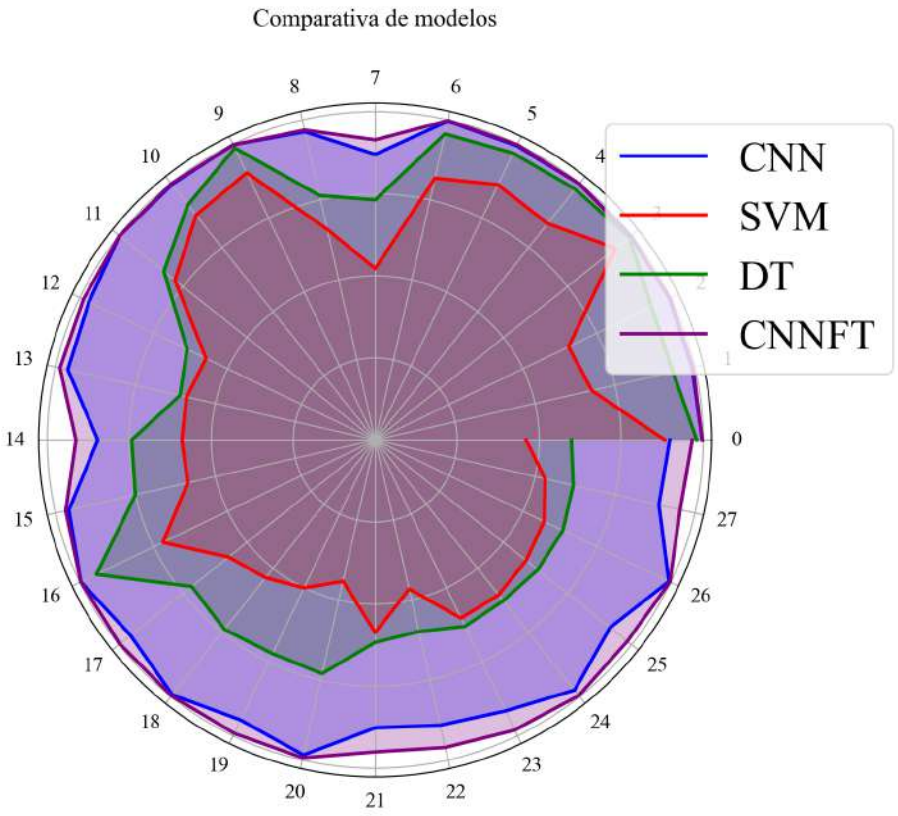


Figura 5.2: Comparativa con gráfico de telaraña.

Simulink. Los modelos simulados se obtuvieron de [1] y corresponden a:

1. Modelo de conmutación de banco de capacitores para generar una señal con un transitorio oscilatorio.

2. Generador de ondas flicker para simular una señal de flicker. La señal extraída es simulada con una frecuencia de flicker de 14 Hz.

3. Modelo de carga monofásico no lineal para generar señales distorsionadas por armónicos.

5.2.1. Modelo de conmutación de banco de capacitores

Este modelo es propuesto por [1] y se compone de una fuente de alimentación de 30 MVA, un transformador el cual disminuye el voltaje de 11kV a 0.4 kV, una carga RL, dos bancos de capacitores y dos interruptores, los cuales al activarse ocasionan la ocsiclación en el sistema. Para la simulación, se utiliza una frecuencia de muestreo de 4000 Hz y una frecuencia nominal de 50 Hz, con lo cual se tienen 80 muestras por ciclo. Esto se hace para que corresponda con la misma frecuencia de muestreo y muestras por ciclo que se utilizan en el entrenamiento descrito en el Capítulo anterior.

La Figura 5.3 muestra el modelo en Simulink que se utiliza para simular una señal que contiene el evento de transitorio oscilatorio correspondiente a la clase {5}, ver Tabla 4.1. La medición utilizada para validar la red neuronal propuesta, corresponde a una señal de voltaje medida en el nodo del lado de baja en el transformador, es decir, el nodo de 0.4 kV, ver Figura 5.3.

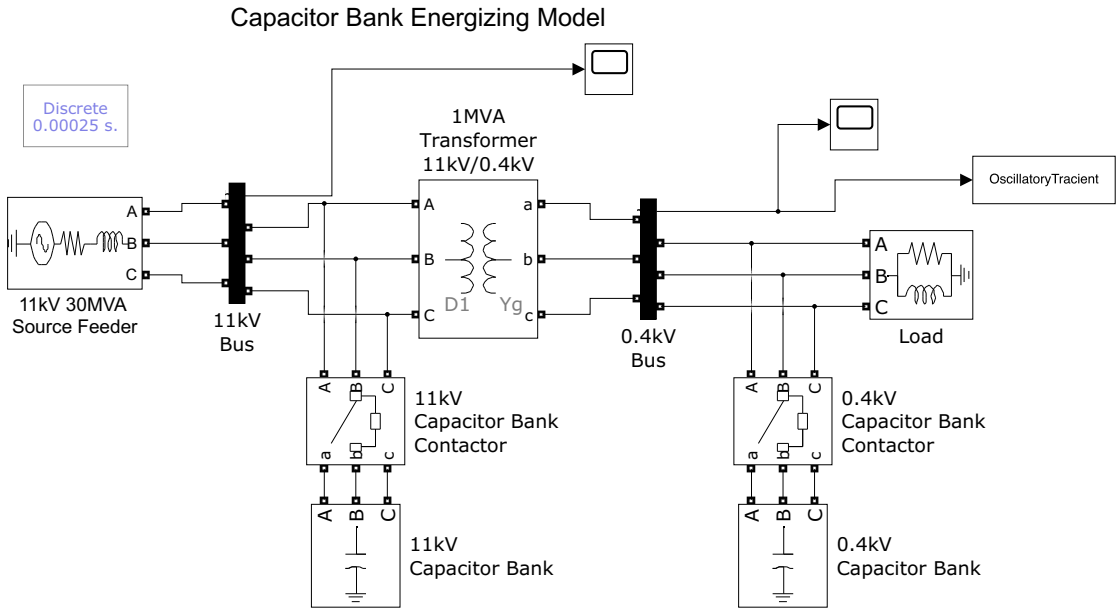


Figura 5.3: Modelo de conmutación de banco de capacitores [1].

La Figura 5.4 muestra la señal de voltaje que se mide en el nodo de 0.4 kV del modelo mostrado en la Figura 5.3. Es importante mencionar que a está señal no se le realiza ningún tipo de pre-procesamiento, por lo que lo único que se utiliza como entrada al algoritmo propuesto, es la señal dividida en ventanas de análisis de 3 ciclos cada una, es decir, la red neuronal mostrada en la Figura 4.9, tendrá como entrada la señal a la cual se desea realizar la identificación y clasificación de eventos PQ, dividida en ventanas de análisis de 3 ciclos de longitud. Así, la red neuronal realiza la identificación y clasificación de los eventos con ese tamaño de ventana. Es por esto, que la señal de voltaje que entra a la red neuronal propuesta es dividida en ventanas, como se muestra en diferentes colores en la Figura 5.4.

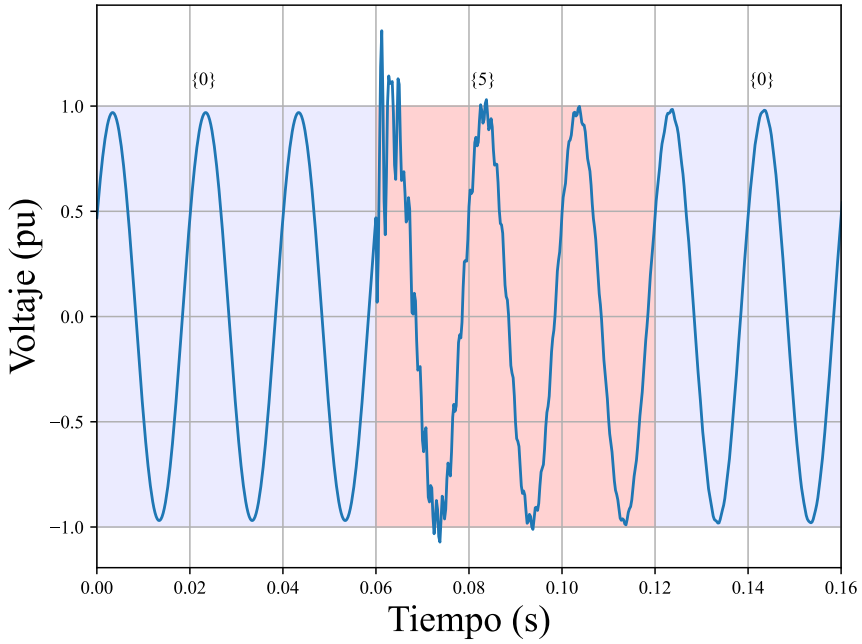


Figura 5.4: Señal de voltaje con transitorio oscilatorio.

De lo anterior, los resultados obtenidos por el modelo de red neuronal propuesta se describen a continuación. La primer ventana corresponde cuando aún no inicia el evento, es decir, los interruptores en la Figura 5.3 están apagados. La segunda ventana corresponde

cuando el evento está presente, es decir, los interruptores se accionan en $t = 0.06$ s, es decir, 3 ciclos después de iniciar la simulación. Por último, la tercer ventana corresponde cuando el evento prácticamente desaparece.

Así, los resultados por arrojados por la red neuronal propuesta, se muestran entre llaves en la parte superior de cada ventana de análisis, donde el número que aparece entre las llaves corresponde a la etiqueta que la red neuronal identifica. De aquí, se observa que logra identificar en la primer ventana, que la señal corresponde a una señal pura, es decir, una señal etiquetada como clase $\{0\}$. En la segunda ventana, la red neuronal identifica la clase transitorio oscilatorio, es decir, una clase $\{5\}$ acorde la Tabla 4.1. Finalmente, para la última ventana, la red nuevamente identifica en la señal de voltaje, una señal pura, es decir, de clase $\{0\}$. Cabe destacar que la red neuronal propuesta identifica y clasifica de manera correcta los 3 casos que aparecen para las 3 ventanas de análisis en las cuales se divide la señal original de voltaje.

5.2.2. Modelo generador de ondas de flicker

A continuación, la Figura 5.5 muestra el modelo propuesto por Rodney Tan en [1] para la simulación de una señal con flicker en un sistema monofásico. El bloque principal de este modelo Simulink, es el bloque Flicker Waveform Generator, donde este bloque se encarga de simular la señal y es el único que se modifica para adaptarlo a las necesidades del algoritmo propuesto.

Este modelo utiliza los mismos parámetros de frecuencia y número de muestras que el modelo anterior. Los parámetros adicionales para este modelo son la frecuencia del flicker y el porcentaje de fluctuación del voltaje, que para este caso de estudio se utiliza una frecuencia de flicker de 16 Hz y una fluctuación de voltaje del 16 %. La medición se realiza en el único nodo que se encuentra en el modelo.

La Figura 5.6 muestra la señal medida del modelo mostrado en la Figura 5.5. Esta señal se divide en 5 ventanas de 3 ciclos cada una y se introducen a la red neuronal

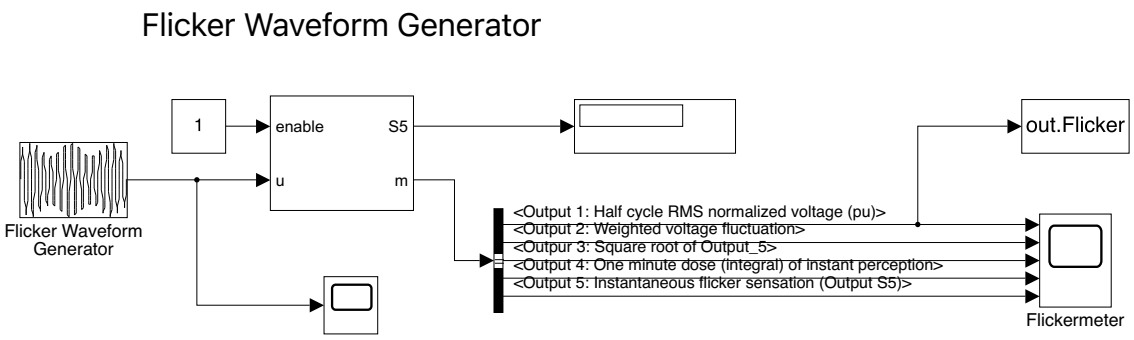


Figura 5.5: Modelo generador de ondas flicker [1].

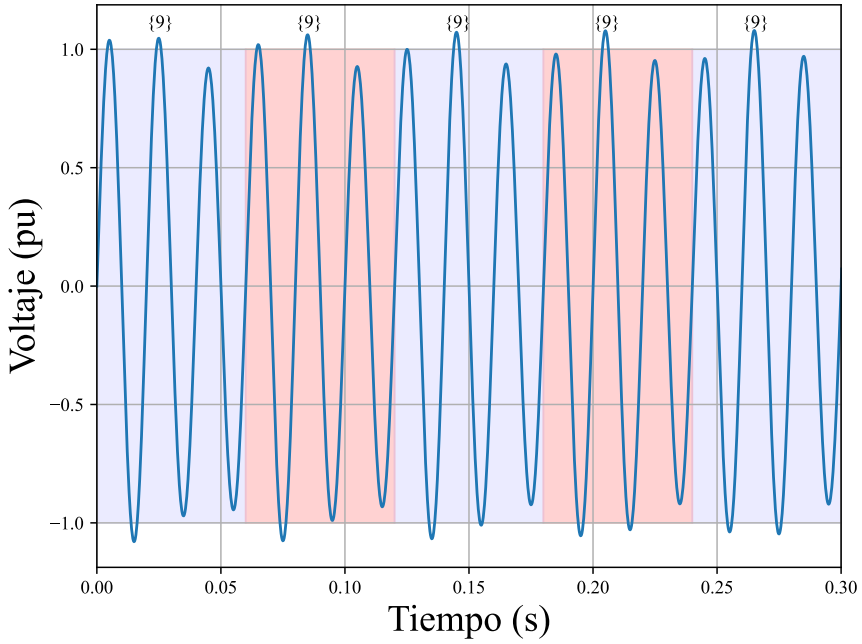


Figura 5.6: Señal de voltaje con flicker.

propuesta de manera similar que en el caso de estudio anterior. La predicción del algoritmo se muestra arriba de cada ventana entre llaves, donde se observa que identifica en todas las ventanas de análisis el flicker, cuya clase es la clase {9}, lo cual es correcto ya que el flicker se encuentra presente durante todo el tiempo de simulación.

5.2.3. Modelo de carga monofásico no lineal

Similarmente a los casos de estudio anteriores, este modelo se simula con una frecuencia de muestro de 4000 Hz, una frecuencia nominal de 50 Hz para tener 80 muestras por ciclo. Los parámetros del modelo original se modificaron de tal forma que, la contaminación armónica fuera más notoria, específicamente para que el THD en la señal de voltaje fuera mayor al 10 %. La Figura 5.7 muestra el modelo de MATLAB/Simulink que se utiliza para generar la señal con contaminación armónica. La señal que se utiliza para validar la red neuronal propuesta corresponde a la señal de voltaje medida en el nodo de 0.4 kV.

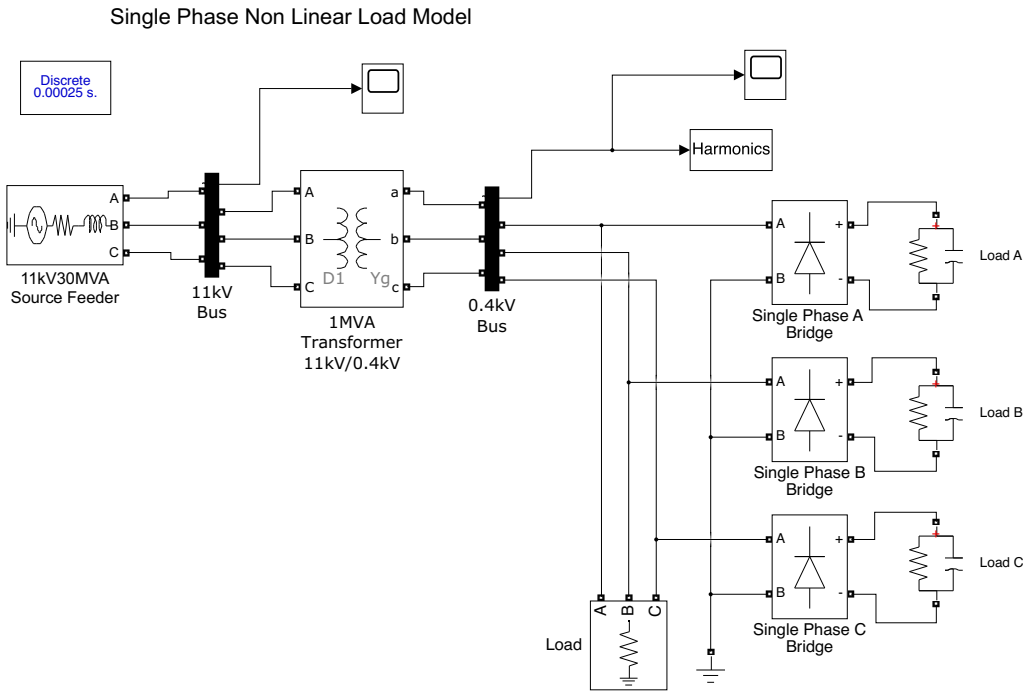


Figura 5.7: Modelo de carga monofásico no lineal [1].

La Figura 5.8 muestra la señal de voltaje del modelo mostrado en la Figura 5.7, la cual es dividida en 4 ventanas de 3 ciclos cada una y se introduce al algoritmo propuesto para identificar a que tipo de clase pertenece la señal. El resultado que arroja la red neuronal se muestra entre llaves ubicadas en la parte de arriba de la señal de voltaje. Así, la clase

predicha por el algoritmo es armónicos, es decir, clase {6} en cada una de las ventanas, lo cual es correcto, ya que el contenido armónico está presente durante todo el tiempo de simulación.

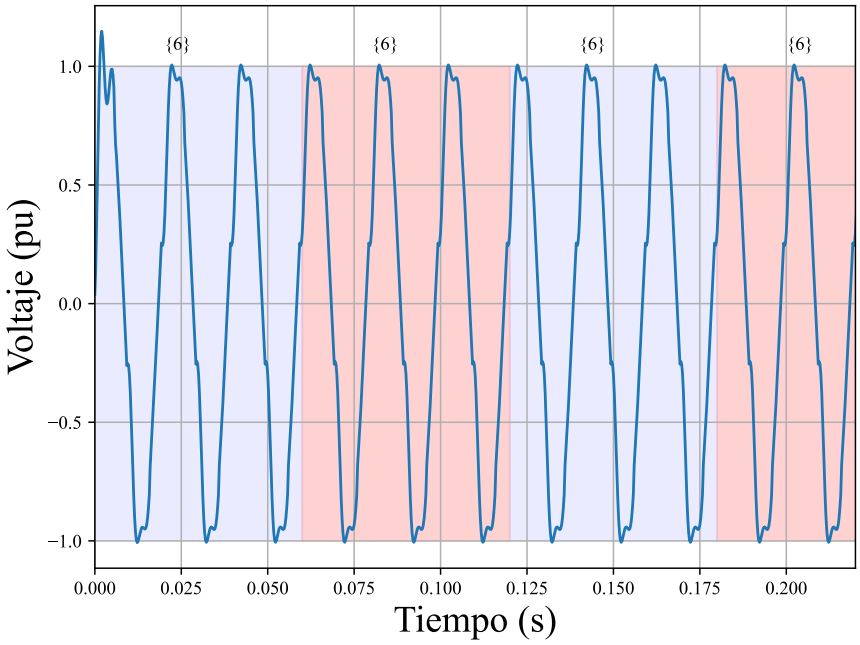


Figura 5.8: Señal de voltaje con contenido armónico.

5.3. Aplicación a señales reales

Esta Sección presenta diferentes conjuntos de datos correspondientes a señales reales de voltaje, las cuales contienen únicamente Sags. Estas señales se pueden descargar directamente de [85]. Así, para validar la red neuronal propuesta se eligen al azar 5 señales con eventos Sags reales de todo el conjunto de señales contenidas en [85]. Estas señales se diseñaron a partir de eventos aislados representativos para una red de 50 Hz con una frecuencia de muestreo de 20 kHz (400 muestras por ciclo) [85]. Por lo tanto, para seguir utilizando la red neuronal propuesta, es necesario acondicionar las señales reales al modelo

de la Figura 4.9, el cual requiere de 80 muestras por ciclo, por lo que es necesario bajar la frecuencia de muestreo a 4000 Hz, esto se logra mediante un submuestreo a las señales.

En esta parte, es importante resaltar que en [85] se menciona que las mediciones de voltaje pertenecen a eventos Sags reales, es decir, que únicamente se presenta un Sag en las señales. Sin embargo, como se muestra más adelante, la red neuronal propuesta es capaz de identificar que los conjuntos de datos utilizados, además de presentar el Sag, también contienen otro tipo de evento PQ, el cual corresponde a la clase {6} de la Tabla 4.1, señal con armónicos. Por lo tanto, para corroborar la efectividad de la red neuronal propuesta, se realiza un espectro de Fourier a las señales reales de prueba, y así corroborar sí contienen o no armónicos.

La Figura 5.9 muestra 33 ciclos de una de las señales reales. Para realizar la validación de la identificación y clasificación de eventos PQ contenidos en la señal, solamente se toma en cuenta una ventana de análisis de 21 ciclos, la cual se muestra en el recuadro de color verde de la Figura 5.9(a). La Figura 5.9(b) muestra el espectro de Fourier de la señal completa, donde se puede observar la presencia de armónicos de 3er, 5to y 7mo orden, respectivamente, lo cual corrobora la presencia de armónicos en la señal real. Por otra parte, la Figura 5.9(c) muestra la ventana de análisis dividida en ventanas de 3 ciclos, las cuales entran al modelo propuesto, donde en la parte superior de la ventana se muestra la clase identificada por la red neuronal propuesta. Por lo que, la red neuronal es capaz de predecir en las primeras tres ventanas los armónicos en la señal, correspondiente a la clase {6}. Posteriormente, en la cuarta ventana identifica un sag con armónicos, es decir, una clase {7}. Después, para la quinta ventana identifica un sag con armónicos y flicker, es decir, una clase {17}. Por último, en las dos últimas ventanas identifica un sag con armónicos, es decir, una clase {7}. Por lo tanto, los resultados obtenidos por la red neuronal respaldan lo reflejado en el espectro de Fourier de la señal original.

De manera similar a la señal real anterior, se procede a analizar el resto de las señales reales. Por lo que, la Figura 5.10(a) muestra 27 ciclos de una segunda señal real,

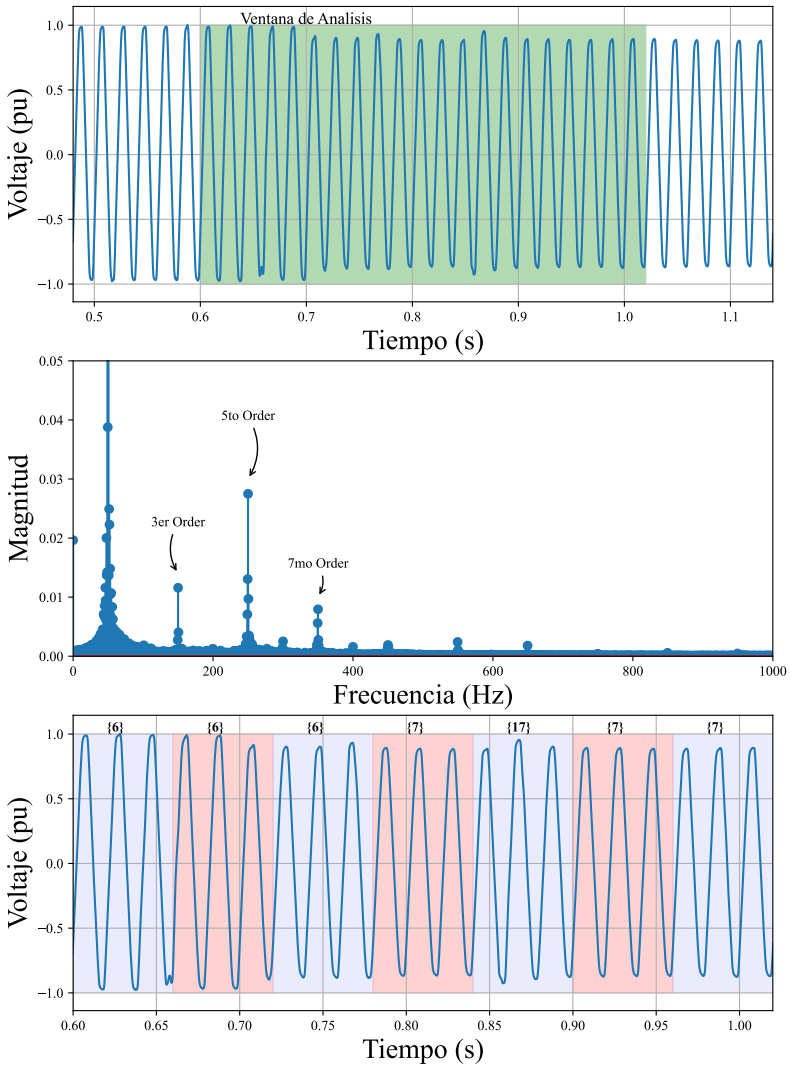


Figura 5.9: Señal real de voltaje 1. (a) Señal completa, (b) espectro de Fourier y (c) resultados de la red neuronal propuesta.

llamada señal real de voltaje 2. Así, La Figura 5.10(b) muestra el espectro de Fourier donde se observa el contenido armónico de la señal, mientras que la Figura 5.10(c) presenta la señal de la ventana de análisis de la Figura 5.10(a) dividida en ventanas de 3 ciclos para su entrada a la red neuronal.

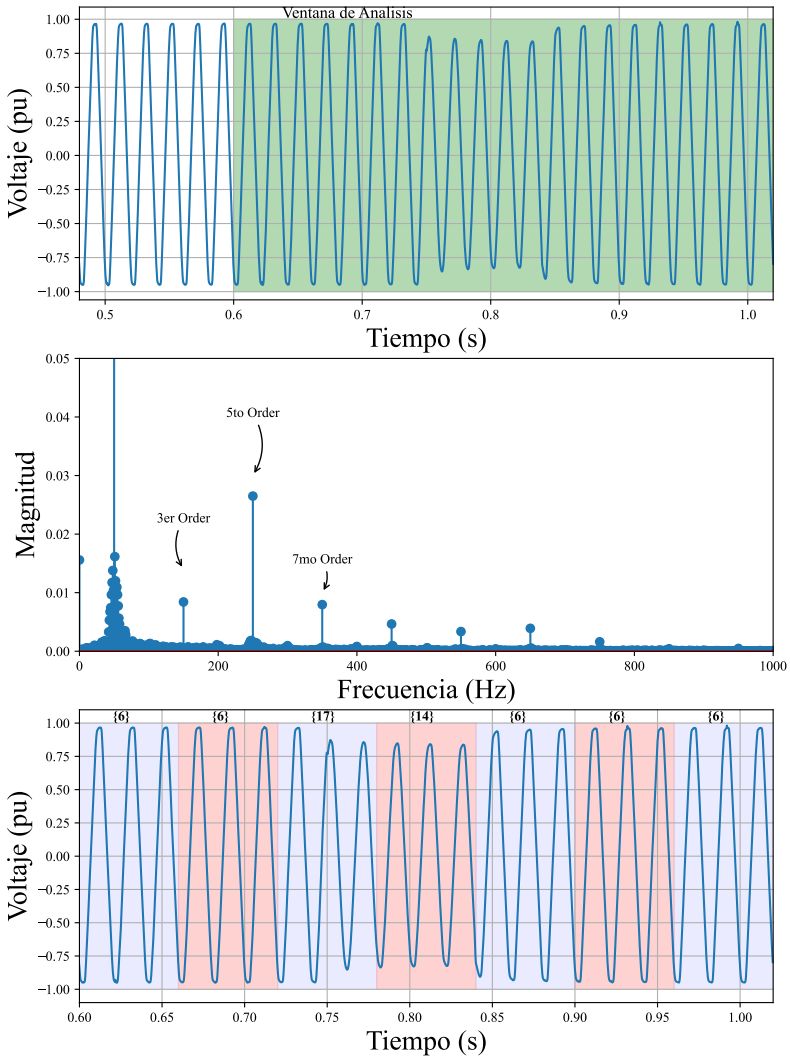


Figura 5.10: Señal real de voltaje 2. (a) Señal completa, (b) espectro de Fourier y (c) resultados de la red neuronal propuesta.

De esta manera, los resultados obtenidos por la red neuronal se presentan para cada ventana de 3 ciclos, donde el modelo propuesto identifica sin problema el contenido armónico presente en toda la señal, correspondiente a la clase {6}. Posteriormente, para la tercera y cuarta ventana, la red neuronal es capaz de identificar contenido armónico con

sag y flicker y sag con armónicos, respectivamente, lo cual corresponde a las clases {17} y {14}, respectivamente.

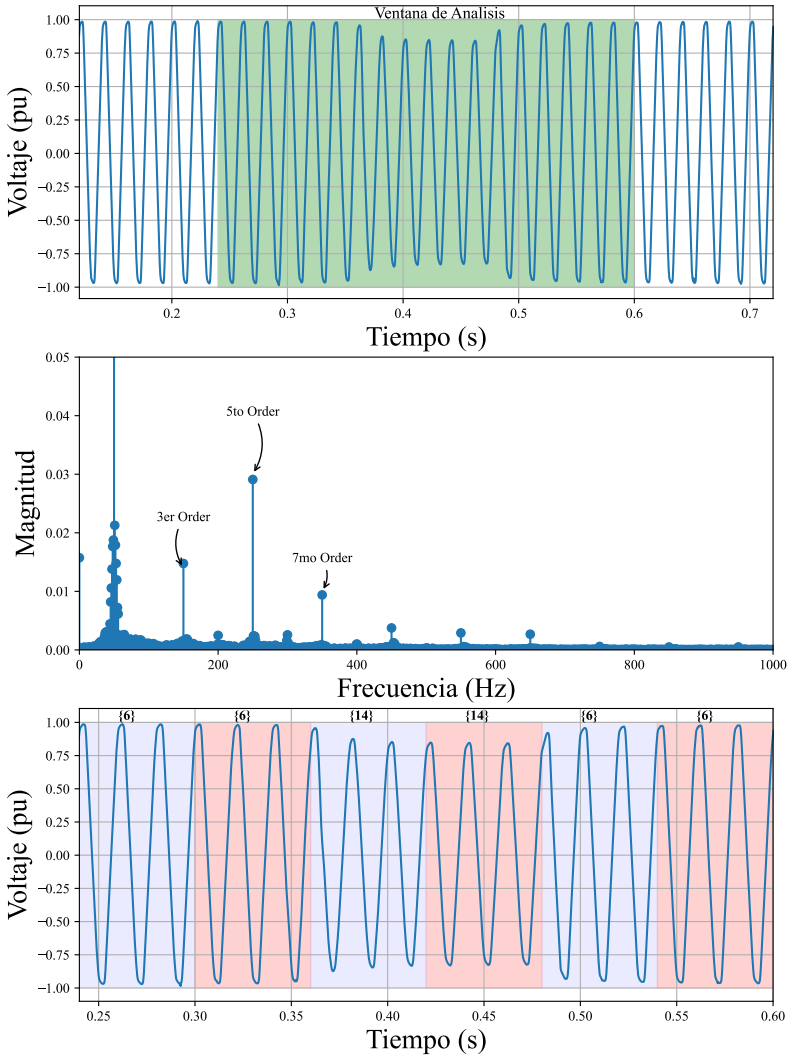


Figura 5.11: Señal real de voltaje 3. (a) Señal completa, (b) espectro de Fourier y (c) resultados de la red neuronal propuesta.

Para una tercer señal real, la Figura 5.11 muestra tanto la señal de voltaje real, así como su correspondiente espectro de Fourier y los resultados arrojados por la red

neuronal propuesta. Se observa que en los resultados presentados en la Figura 5.11(c), la red neuronal identifica el contenido armónico presente en toda la señal, lo cual es validado por el espectro de Fourier de la Figura 5.11(b) y que corresponde a la clase {6}. Por otro lado, el algoritmo también logra identificar el sag con armónicos en la tercera y cuarta ventanas correspondiente a la clase {14}, con lo cual se valida la propuesta.

Para la cuarta señal real analizada, la Figura 5.12(a) muestra un fragmento de 30 ciclos de la señal, así como su respectivo espectro de Fourier, Figura 5.12(b), y la misma señal dividida en ventanas de 3 ciclos para su análisis mediante el algoritmo propuesto. Así, los resultados arrojados para cada ventana de 3 ciclos son: para las primeras 2 y últimas 2 ventanas se predice como clase {6}, es decir, armónicos. Mientras que para la 3ra ventana se predice como clase {14}, sag con armónicos; y para la 5ta ventana como sag con flicker y armónicos, es decir, clase {19}. Por lo tanto, se observa que la red neuronal identifica de manera correcta el sag presente en la señal real, así como el contenido armónico presente en toda la señal y un flicker en cierto periodo de tiempo.

Finalmente, la Figura 5.13(a) muestra 39 ciclos para una quinta y última señal analizada con la red neuronal propuesta. Así, la Figura 5.13(b) muestra el espectro de Fourier correspondiente a la señal, mientras que la Figura 5.13(c) ilustra la señal divida en ventanas que se introducen al algoritmo propuesto y su predicción realizada, donde se detecta apropiadamente el sag en la 4ta y 5ta ventana y detectando armónicos en cada una de las ventanas, es decir, clases {6} y {14}, respectivamente.

Con estos resultados, se concluye la etapa experimental, demostrando que la red neuronal logra obtener resultados adecuados en los tres tipos de señales mostradas, las cuales son: sintéticas, simuladas y reales.

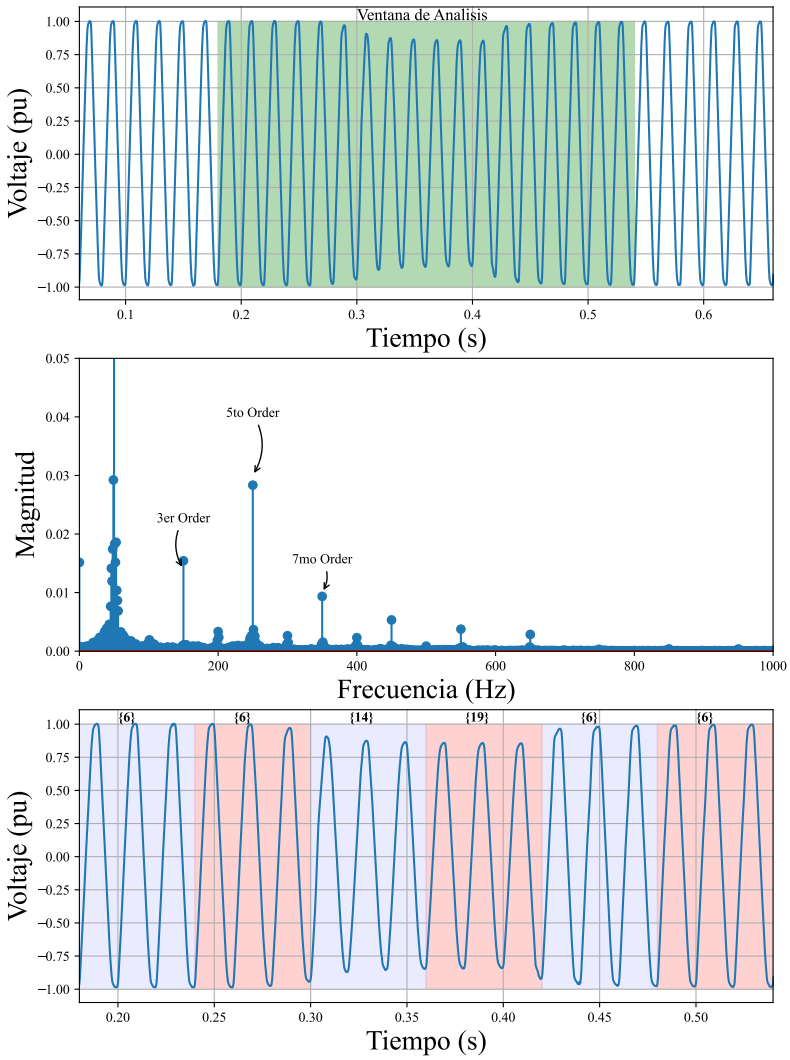


Figura 5.12: Señal real de voltaje 4. (a) Señal completa, (b) espectro de Fourier y (c) resultados de la red neuronal propuesta.

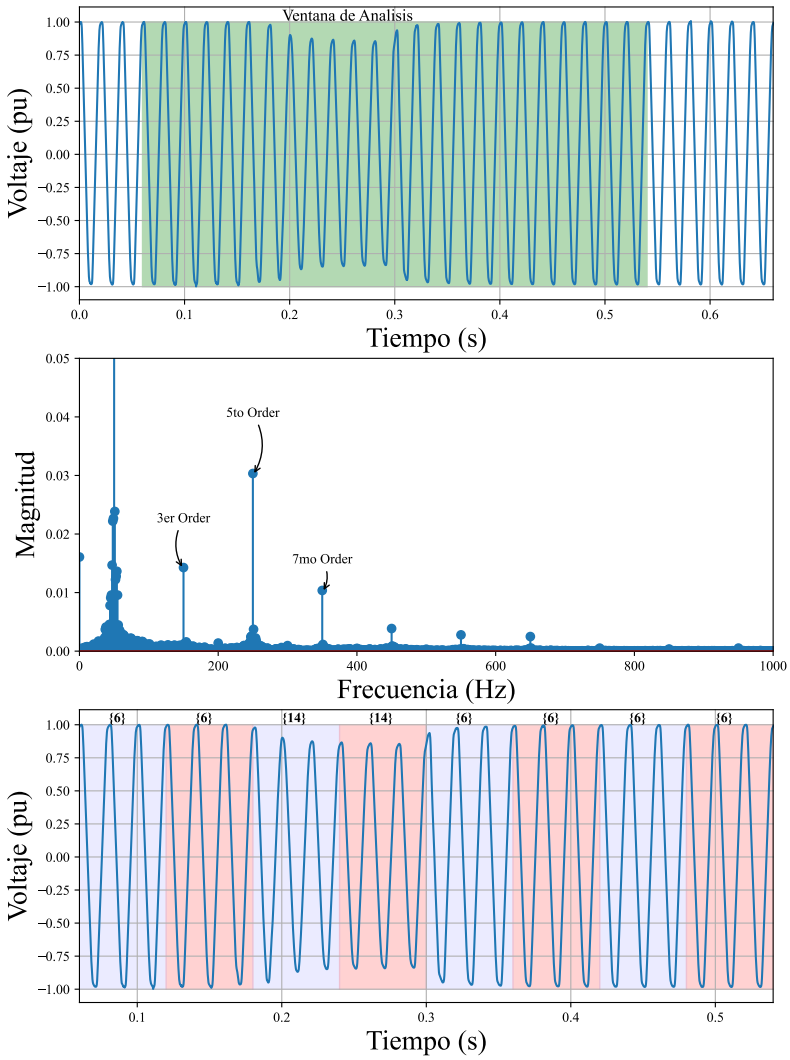


Figura 5.13: Señal real de voltaje 5. (a) Señal completa, (b) espectro de Fourier y (c) resultados de la red neuronal propuesta.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones generales

Se logra demostrar que al combinar las técnicas de aprendizaje por transferencia y sintonización fina con una red neuronal convolucional, la identificación y clasificación de eventos en calidad de la energía es sumamente eficiente para todas las señales sintéticas anteriormente presentadas, así como para señales tanto simuladas como reales. Por lo que una posible implementación en tiempo real es algo que puede ser viable dado el costo computacional y eficiencia del algoritmo.

Por otra parte, durante este trabajo se evidencia lo difícil que es extraer y/o interpretar las características aprendidas por el modelo, características que son muy valiosas al aplicar las técnicas de aprendizaje profundo pero que son aún más valiosas si se pudieran interpretar.

Una ventaja a destacar, es la precisión, robustez y versátilidad del modelo propuesto, ya que el modelo puede clasificar hasta 28 señales con combinaciones distintas de los eventos más comunes que ocurren en la calidad de la energía, siendo esto una contribución al estado del arte actual para esta área de estudio.

Otra de las ventajas importantes de este modelo es que la etapa de preprocesa-

miento no es necesaria, ya que la red neuronal convolucional realiza de manera interna la extracción de características, por lo que no se requiere una etapa de preprocesamiento adicional, además, el costo computacional de esta extracción de características solo se genera durante el entrenamiento, es decir, que durante la predicción del evento no se estaría requiriendo este recurso computacional.

Algo que se puede catalogar como una desventaja, es el tiempo de preparación del modelo, es decir, el tiempo de entrenamiento del algoritmo, desde la etapa de preentrenamiento de los modelos hasta la etapa de validación en la sintonización fina, esto podría ser una desventaja al intentar actualizar el modelo con una señal totalmente nueva o al intentar agregar una característica especial o especifica en una señal, ya que se tendrían que realizar el entrenamiento desde la etapa de sintonizacion fina.

6.2. Trabajos futuros

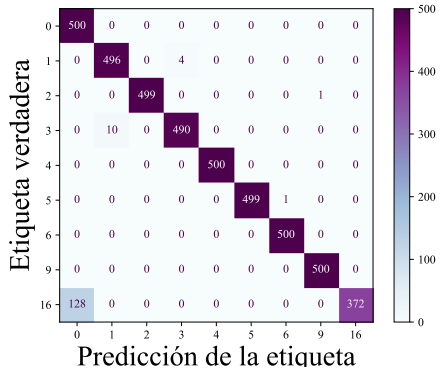
Como posibles futuro trabajos se tienen los siguientes:

Reducir u optimizar el tamaño del modelo propuesto a través de algoritmos de compresión para redes neuronales, este trabajo consistiria en hacer el modelo mas pequeño conservando su eficiencia. Aplicar un modelo similar para la identificación de fallas eléctricas en SEPs en lugar de eventos PQ, es decir, realizar el mismo desarrollo pero utilizando señales que representen fallas eléctricas. Reducir las etapas de aprendizaje del modelo. Hacer el modelo aún más robusto al agregar distintos tipos de armónicas, inter-armónicas, sub-armónicas y/o supra-armónicas, es decir entrenar el modelo con una variedad mas amplia de armónicas y sus caracteristicas especiales. Implementación del modelo propuesto en tiempo real, llevar al modelo a un esenario real, al programarlo dentro de un PMU o algún medidor de calidad de la energía.

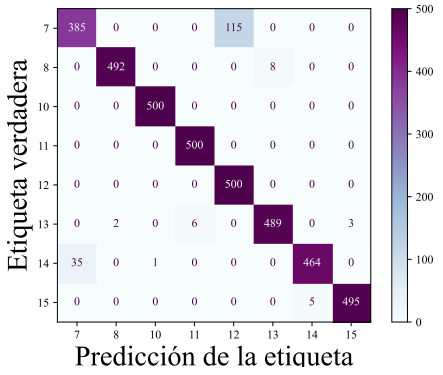
Apéndice A

Creación de modelos usando diferentes número de ciclos

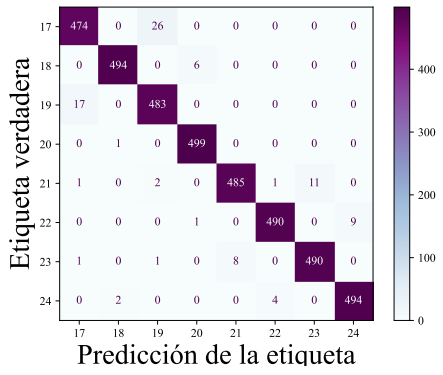
En esta sección se agrega como referencia todos los resultados de los modelos pre-entrenados y los modelos con aprendizaje por transferencia que se crearon a partir de estos, para la creación de la Tabla 4.2 y que es utilizada para obtener el modelo más óptimo.



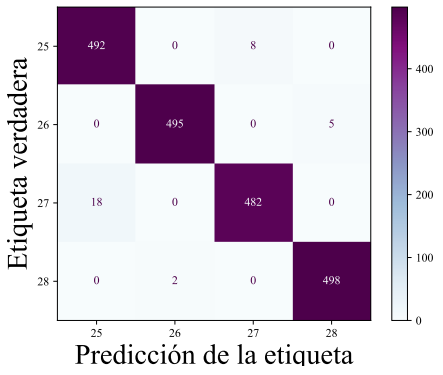
(a) ***Eventos simples***



(b) ***Señales complejas con hasta dos eventos***



(c) ***Señales complejas con hasta tres eventos***



(d) ***Señales complejas con hasta cuatro eventos***

Figura A.1: Resultados para el modelo con entrada de 1 ciclo.

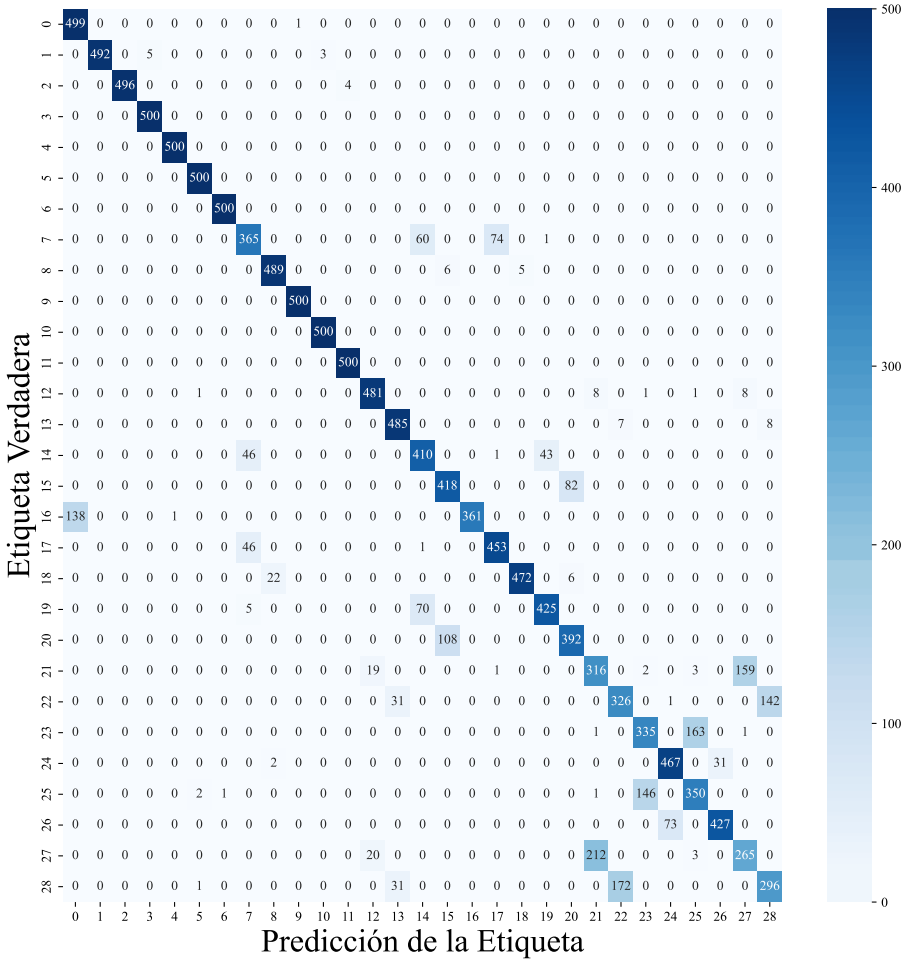
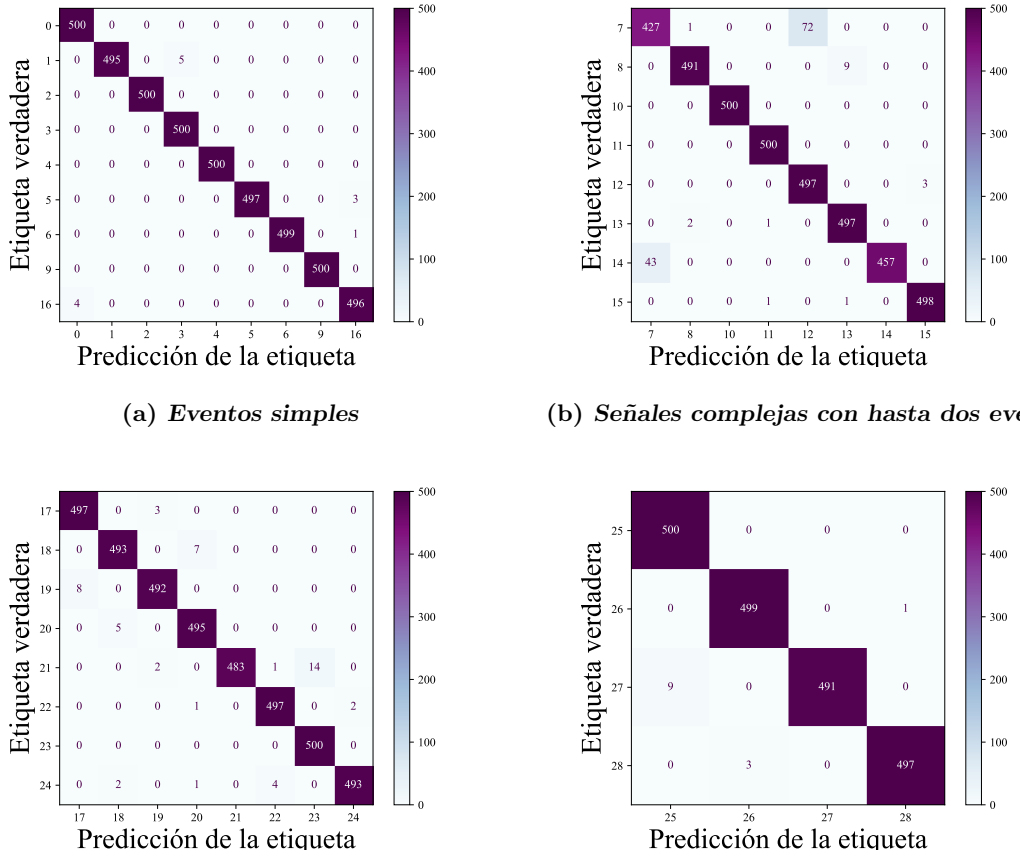


Figura A.2: Resultados del modelo con entrada de 1 ciclo después de SF.



(a) ***Eventos simples*** **(b)** ***Señales complejas con hasta dos eventos***

(c) ***Señales complejas con hasta tres eventos*** **(d)** ***Señales complejas con hasta cuatro eventos***

Figura A.3: Resultados modelo con entrada de 2 ciclos

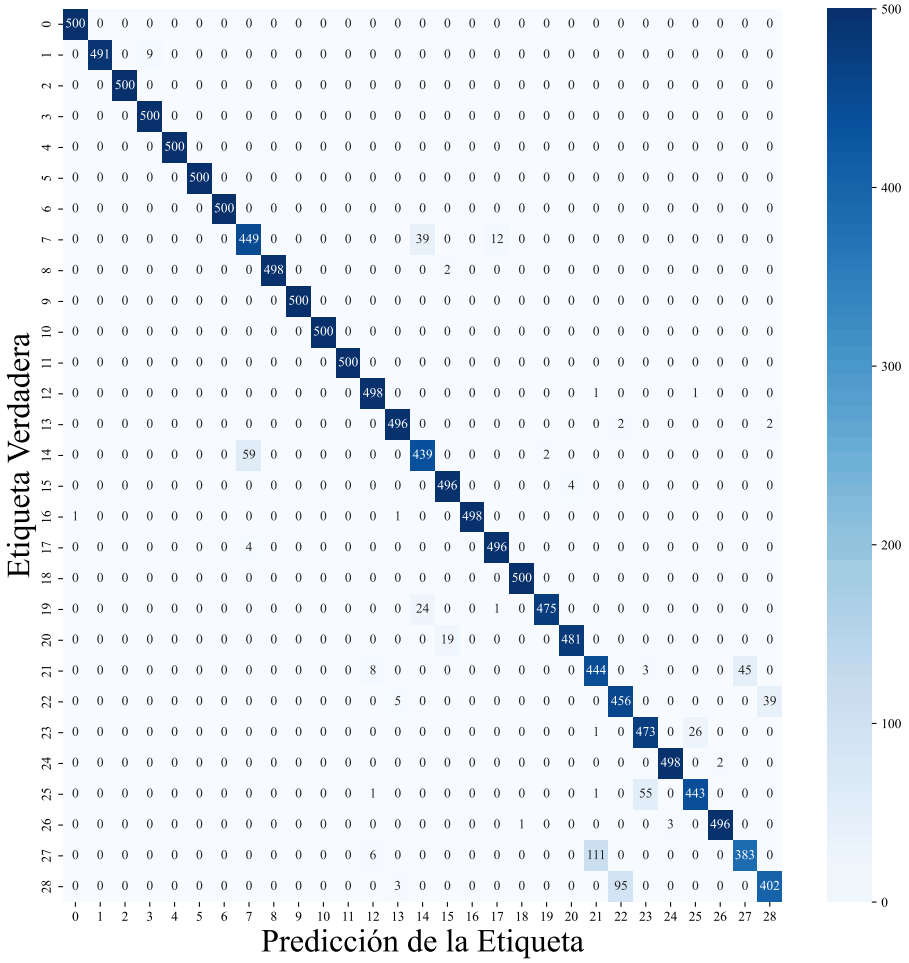
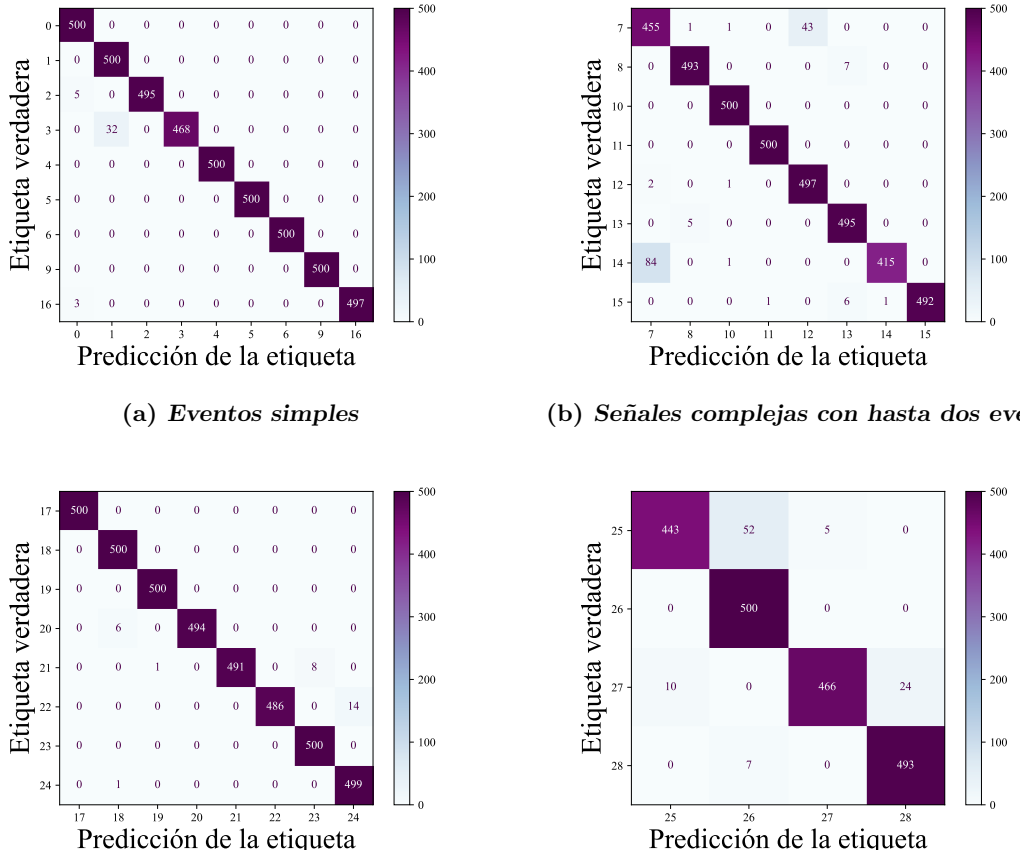


Figura A.4: Resultados del modelo con entrada de 2 ciclos después de SF



(a) ***Eventos simples*** **(b)** ***Señales complejas con hasta dos eventos***

(c) ***Señales complejas con hasta tres eventos*** **(d)** ***Señales complejas con hasta cuatro eventos***

Figura A.5: Resultados modelo con entrada de 4 ciclos

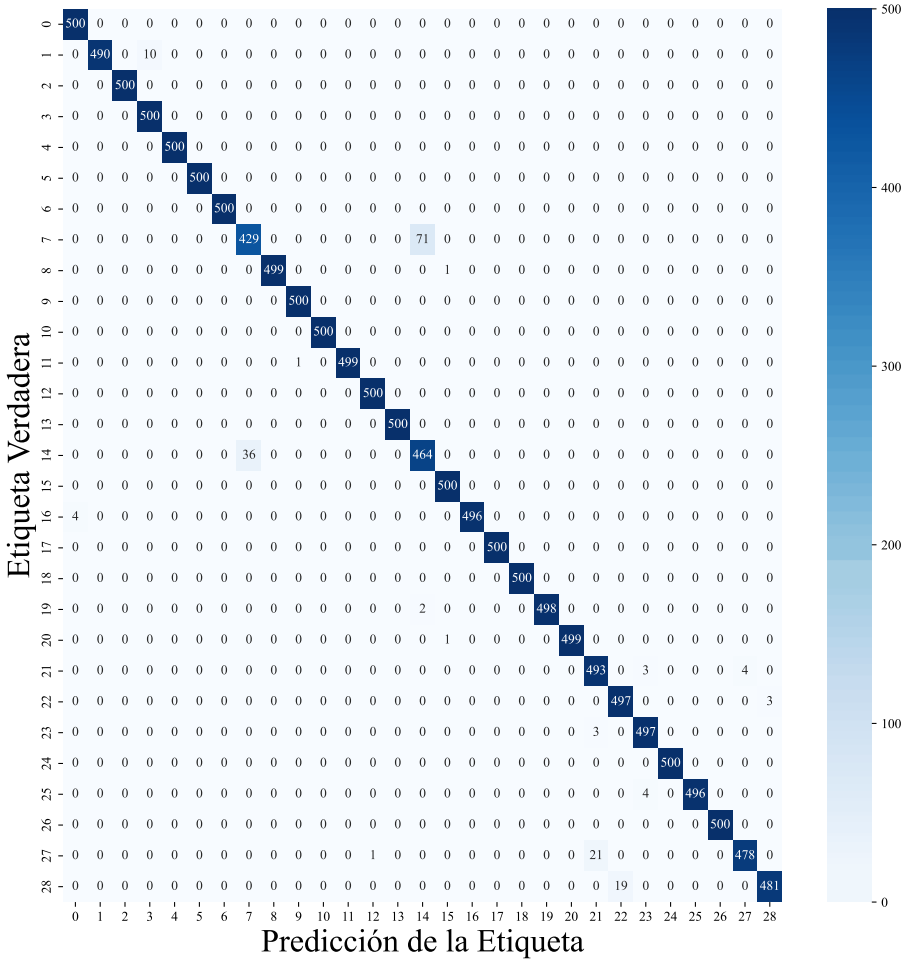
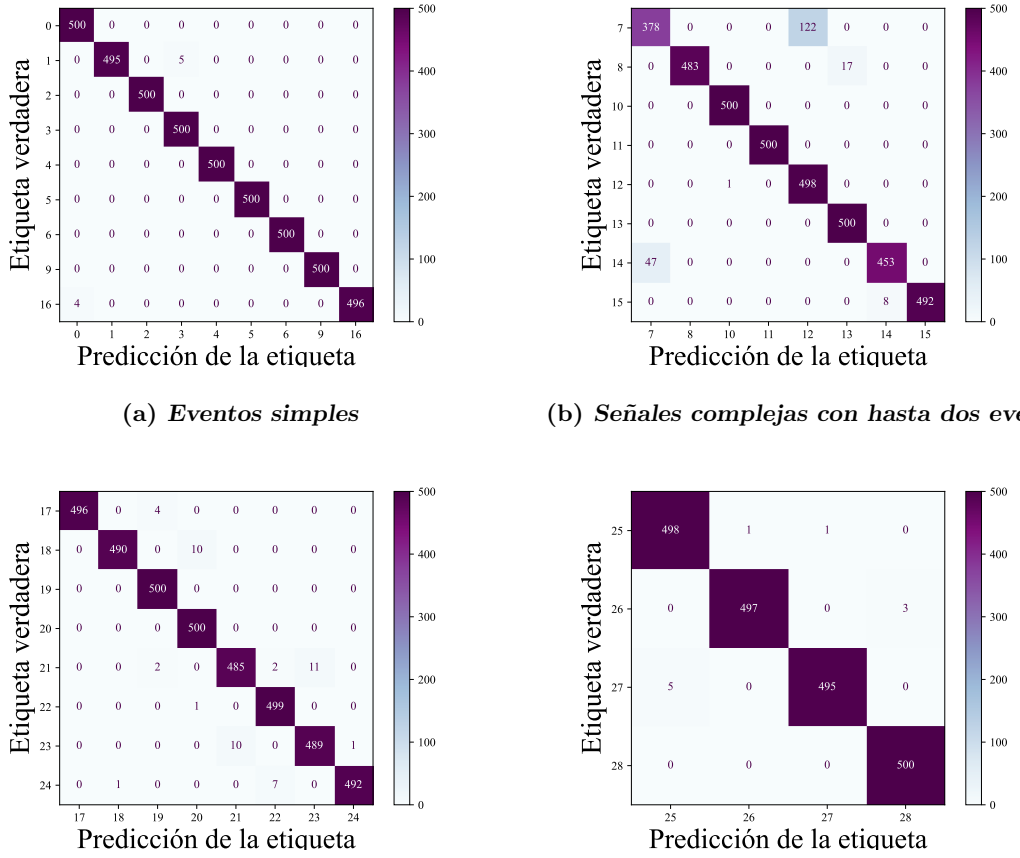


Figura A.6: Resultados del modelo con entrada de 4 ciclos después de SF



(a) ***Eventos simples*** **(b)** ***Señales complejas con hasta dos eventos***

(c) ***Señales complejas con hasta tres eventos*** **(d)** ***Señales complejas con hasta cuatro eventos***

Figura A.7: Resultados modelo con entrada de 5 ciclos

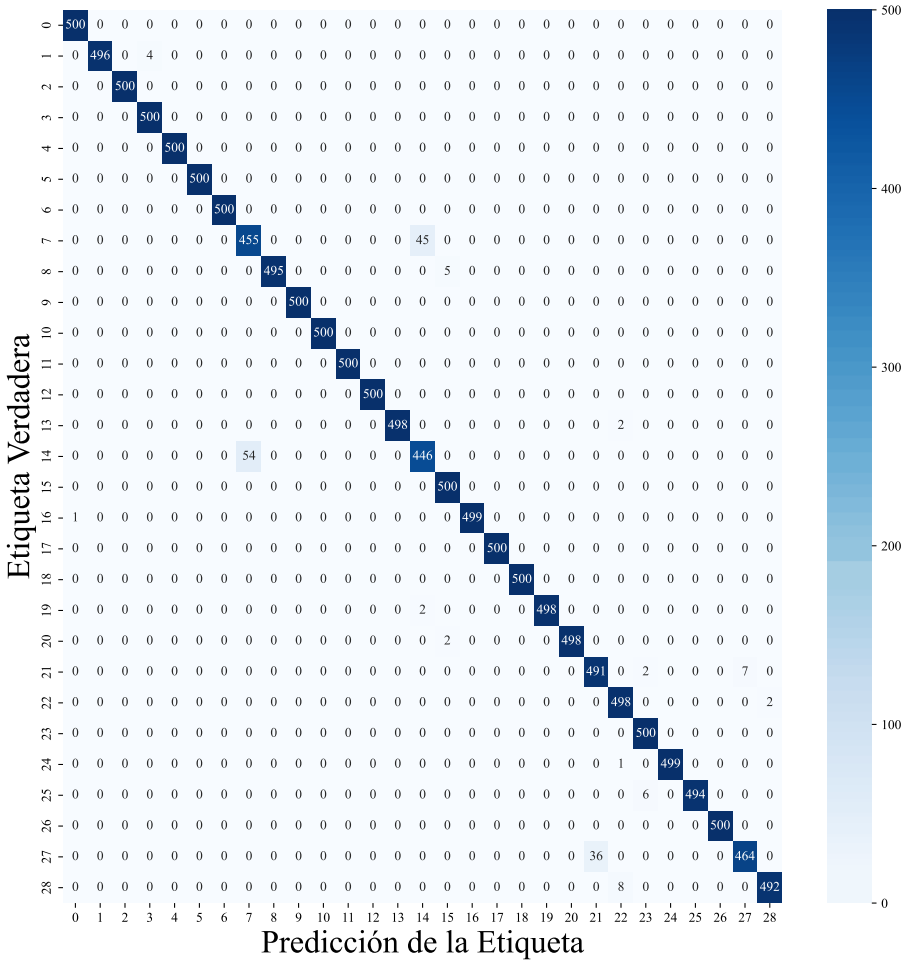
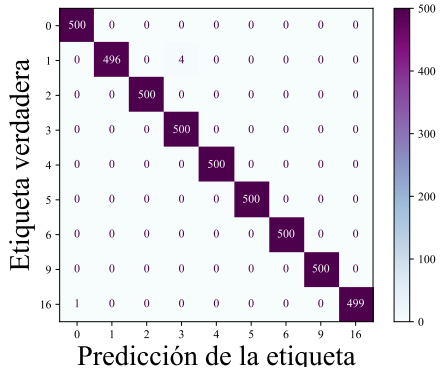
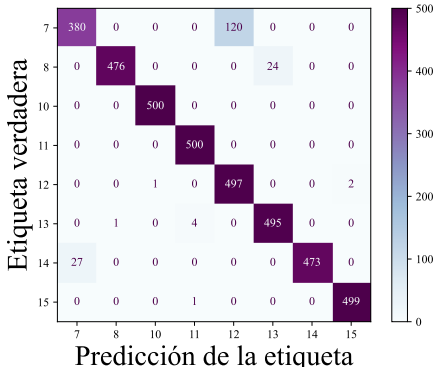


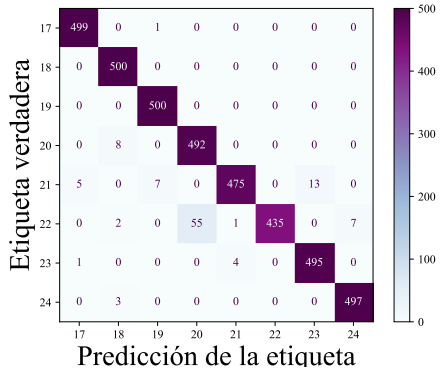
Figura A.8: Resultados del modelo con entrada de 5 ciclos después de SF



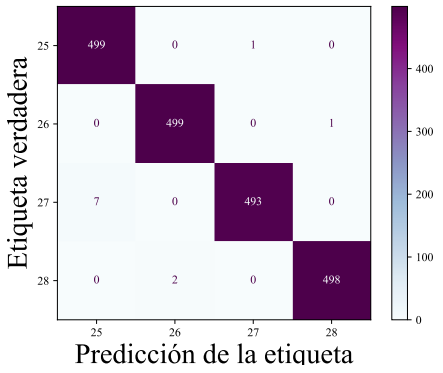
(a) ***Eventos simples***



(b) ***Señales complejas con hasta dos eventos***



(c) ***Señales complejas con hasta tres eventos***



(d) ***Señales complejas con hasta cuatro eventos***

Figura A.9: Resultados modelo con entrada de 6 ciclos

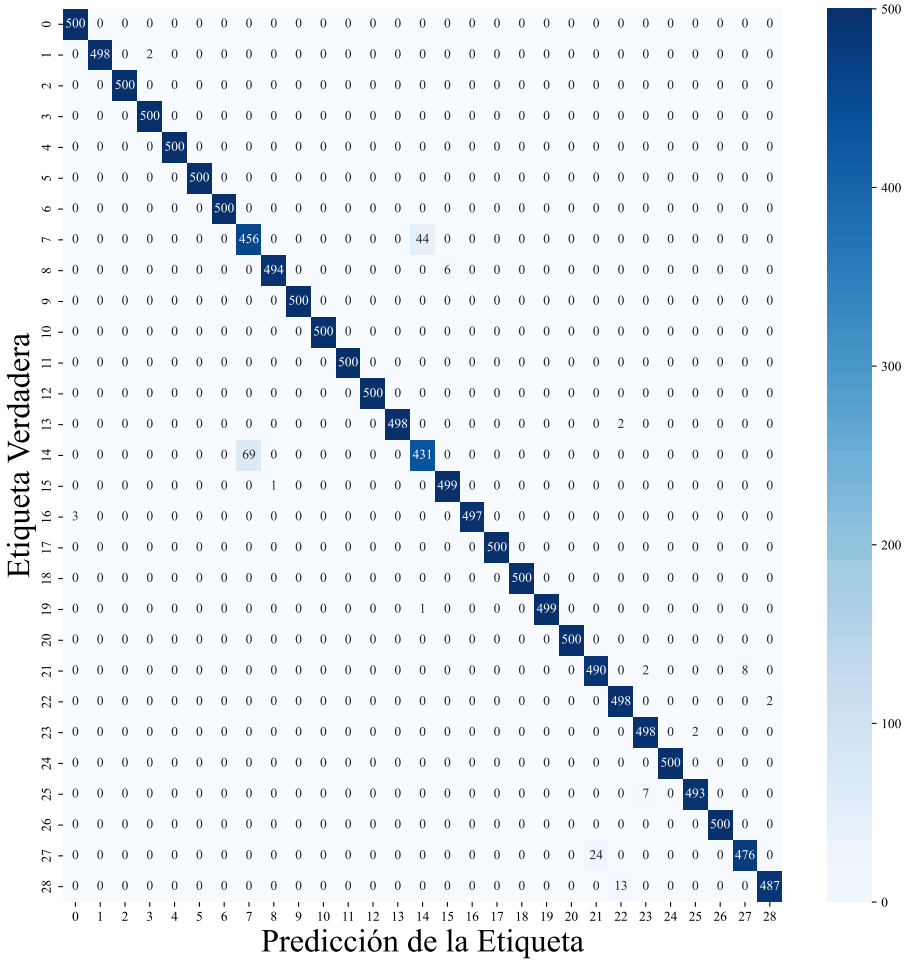
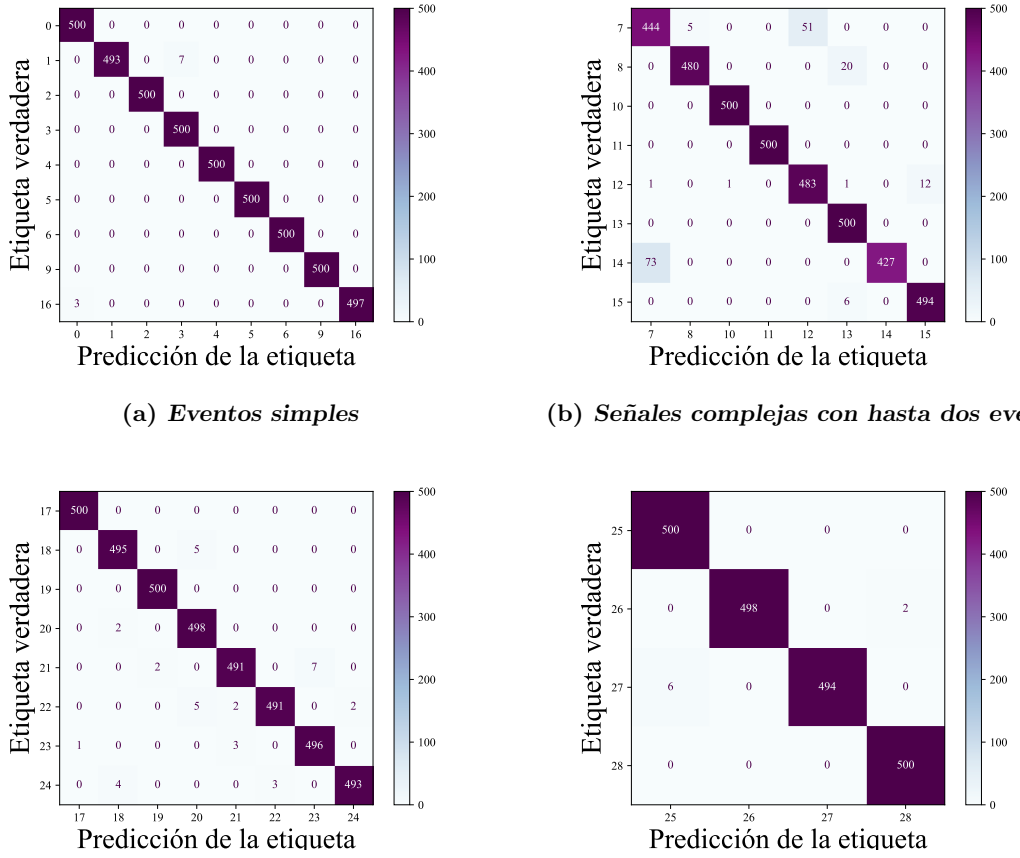


Figura A.10: Resultados del modelo con entrada de 6 ciclos después de SF



(a) ***Eventos simples*** **(b)** ***Señales complejas con hasta dos eventos***

(c) ***Señales complejas con hasta tres eventos*** **(d)** ***Señales complejas con hasta cuatro eventos***

Figura A.11: Resultados modelo con entrada de 7 ciclos

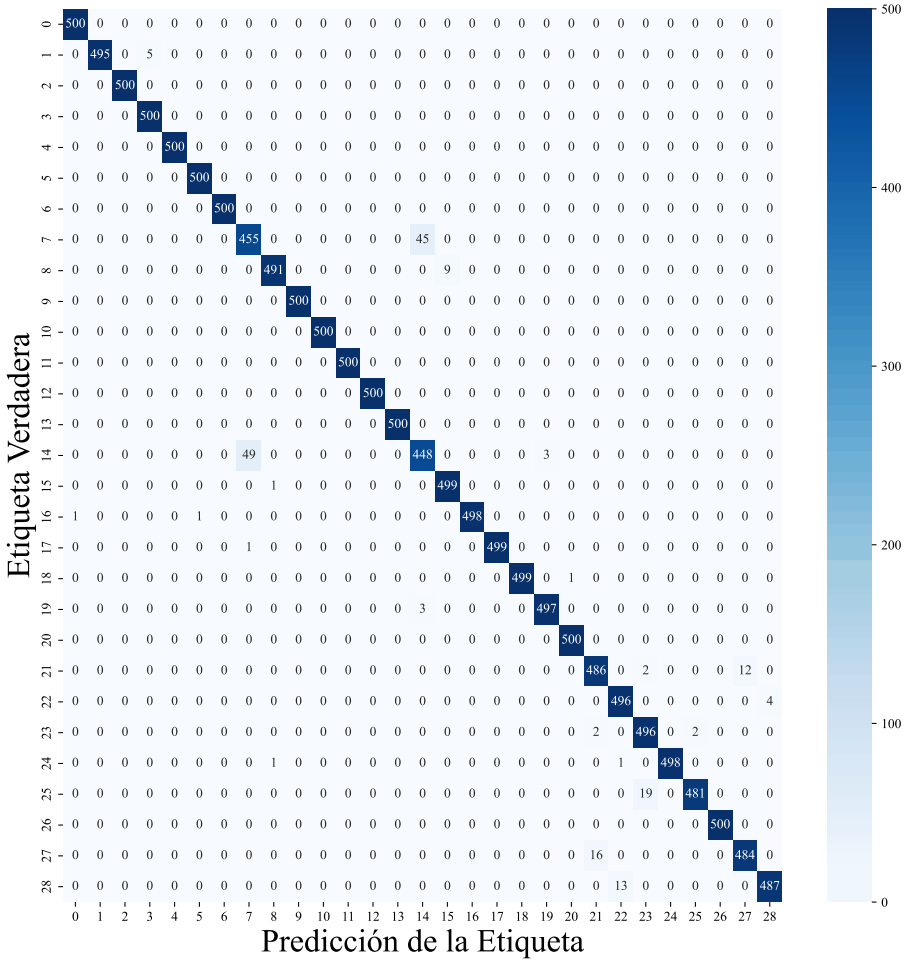
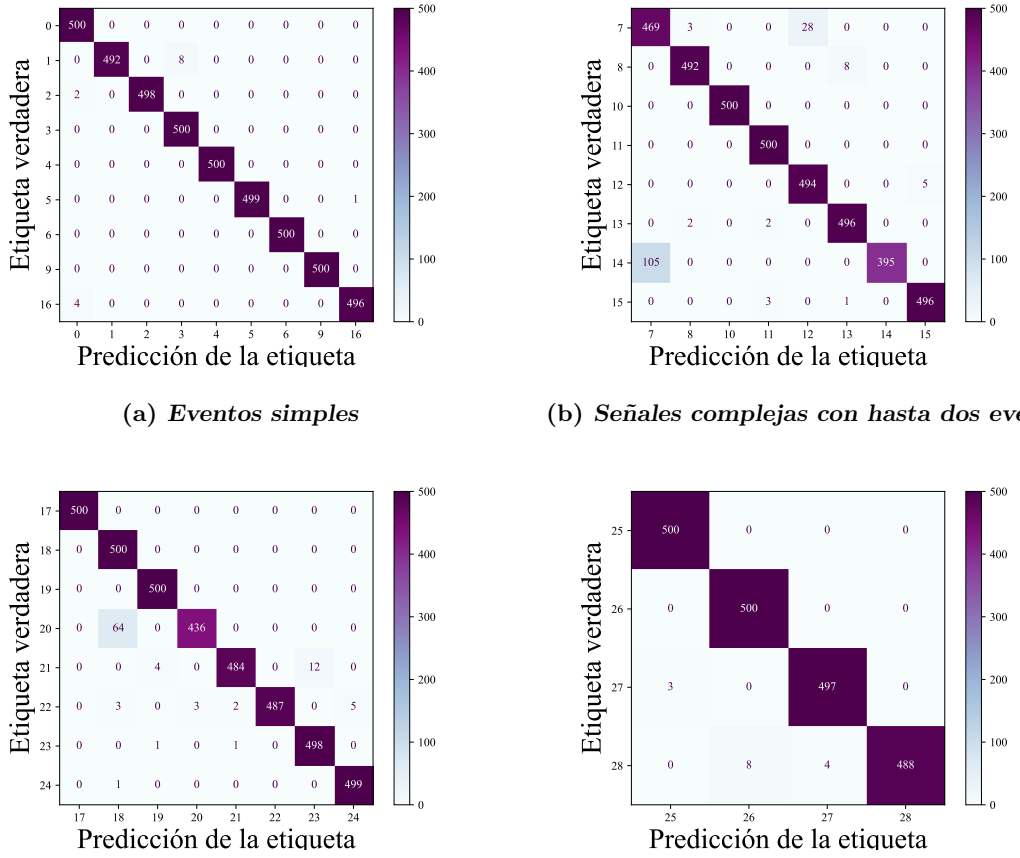


Figura A.12: Resultados del modelo con entrada de 7 ciclos después de SF



(a) ***Eventos simples*** **(b)** ***Señales complejas con hasta dos eventos***

(c) ***Señales complejas con hasta tres eventos*** **(d)** ***Señales complejas con hasta cuatro eventos***

Figura A.13: Resultados modelo con entrada de 8 ciclos

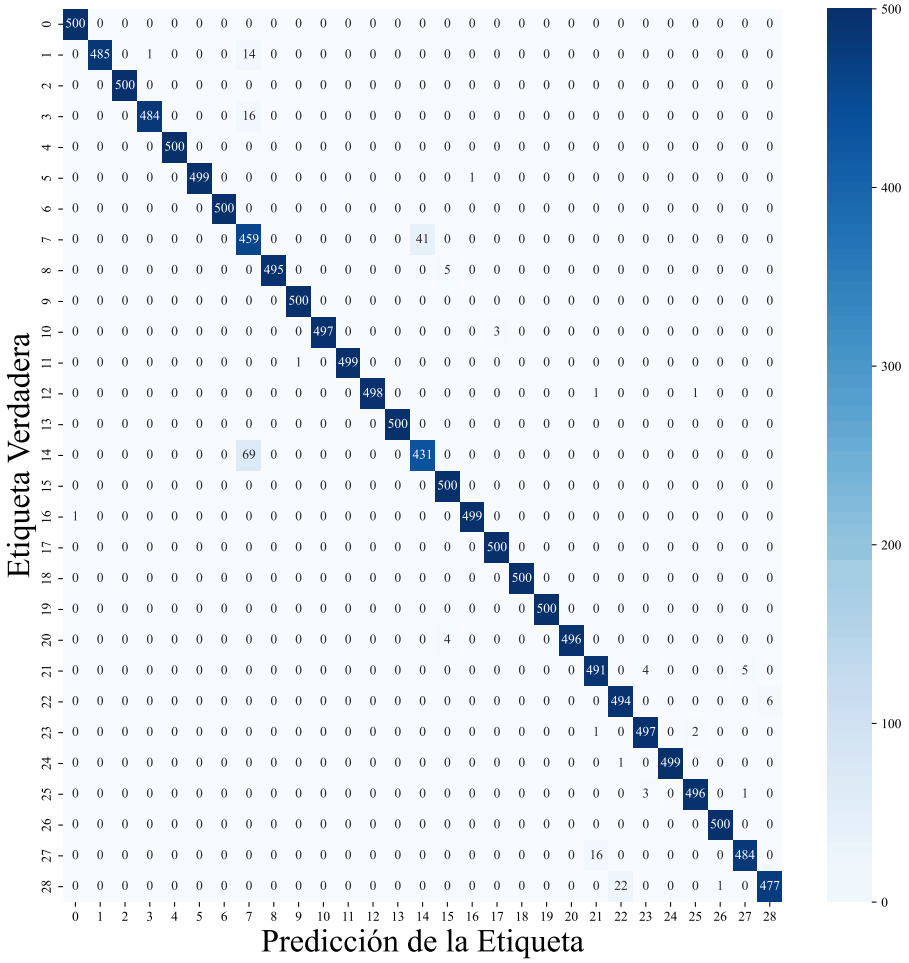
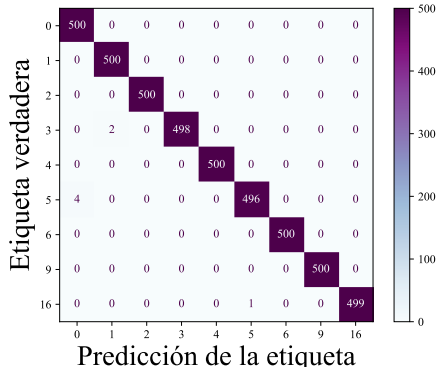
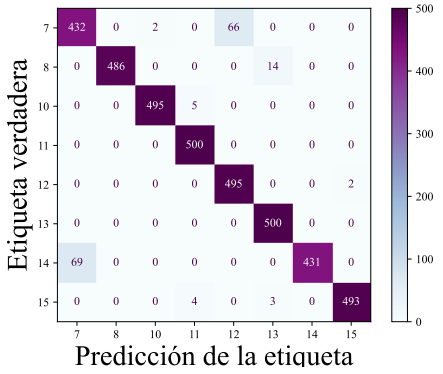


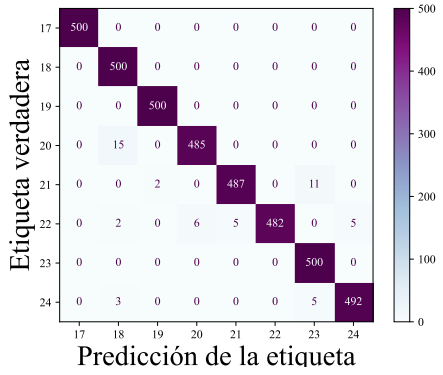
Figura A.14: Resultados del modelo con entrada de 8 ciclo después de SF



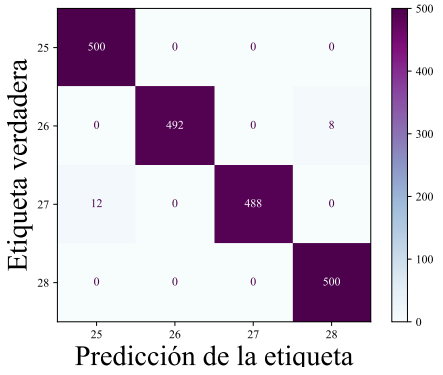
(a) ***Eventos simples***



(b) ***Señales complejas con hasta dos eventos***



(c) ***Señales complejas con hasta tres eventos***



(d) ***Señales complejas con hasta cuatro eventos***

Figura A.15: Resultados modelo con entrada de 9 ciclo

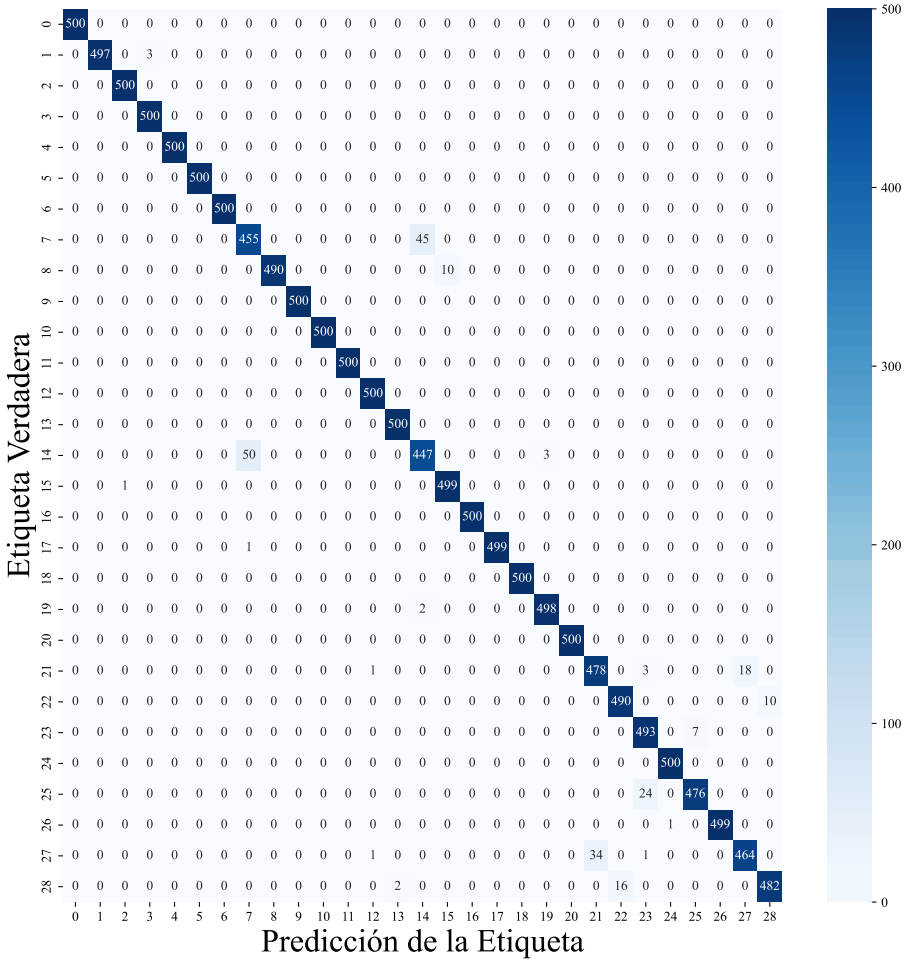
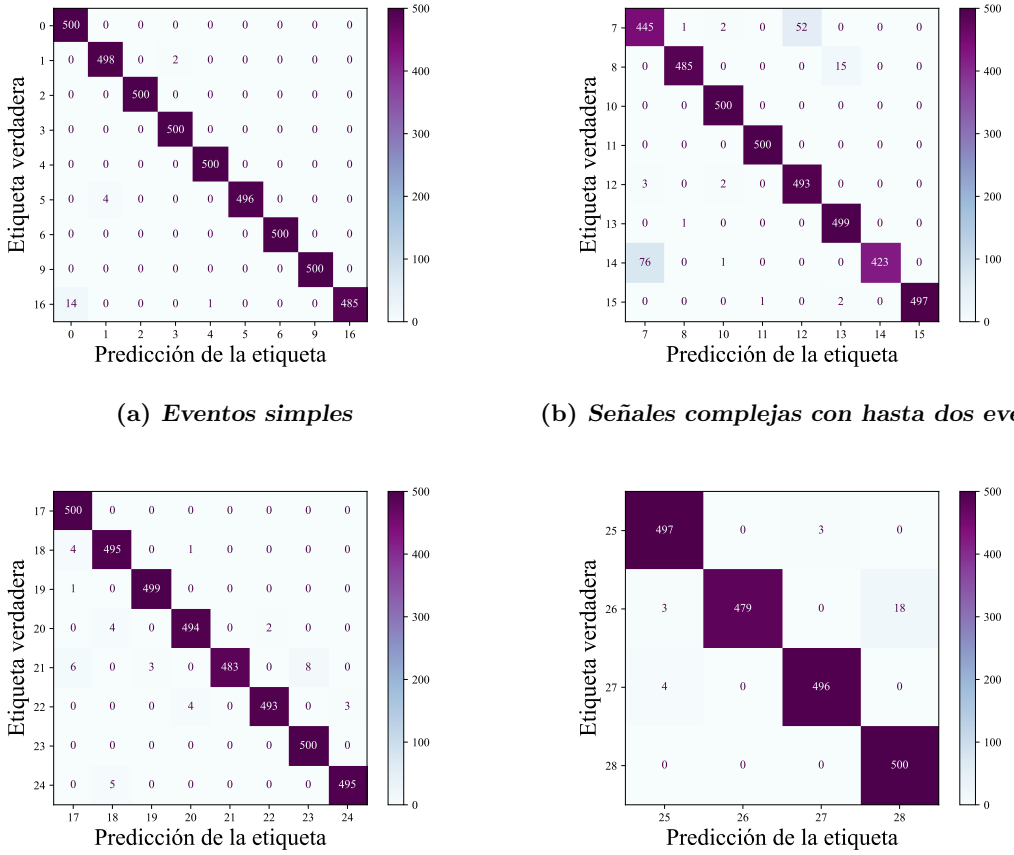


Figura A.16: Resultados del modelo con entrada de 9 ciclos después de SF



(a) ***Eventos simples*** **(b)** ***Señales complejas con hasta dos eventos***

(c) ***Señales complejas con hasta tres eventos*** **(d)** ***Señales complejas con hasta cuatro eventos***

Figura A.17: Resultados modelo con entrada de 10 ciclos

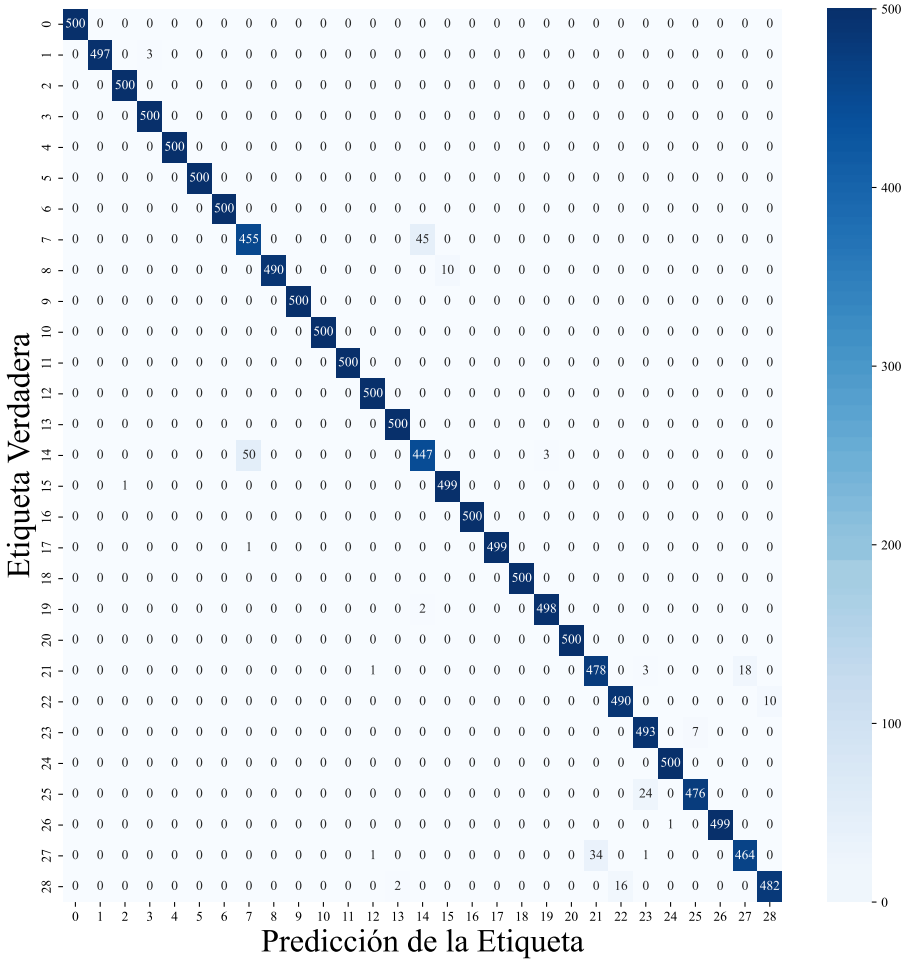


Figura A.18: Resultados del modelo con entrada de 10 ciclos después de SF

Referencias

[1] R. H. Tan and V. K. Ramachandaramurthy, "A comprehensive modeling and simulation of power quality disturbances using matlab/simulink," in *Power Quality Issues in Distributed Generation* (J. Luszcz, ed.), ch. 3, p. N/A, Rijeka: IntechOpen, 2015.

[2] International Electrotechnical Commission (IEC), *IEC 61400-21: Wind turbine generator systems Measurement of power quality characteristics*, 2001. Accessed: 2024-12-04.

[3] IEEE, *IEEE 1159 2009: Recommended Practice for Monitoring Electric Power Quality*, 2009. Accessed: 2024-12-04.

[4] E. C. for Standardization (CEN), *EN 50160: Voltage characteristics of electricity supplied by public distribution systems*, 2010. Accessed: 2024-12-04.

[5] IEEE, *IEEE 1459: Standard Definitions for the Measurement of Power Quality*, 2015. Accessed: 2024-12-04.

[6] IEEE, *IEEE 929 2000: Standard for Utility Interface of Photovoltaic (PV) Systems*, 2000. Accessed: 2024-12-04.

[7] I. E. Commission, "Iec 61000-4-30: 2015 - electromagnetic compatibility (emc) - part 4-30: Testing and measurement techniques - power quality measurement methods," 2015. IEC.

[8] S. A. Soliman and A. M. Alkandari, "Electric power systems harmonics identification and measurements," in *Power Quality* (G. R. Rey and L. M. Muneta, eds.), ch. 1, Rijeka: IntechOpen, 2011.

[9] v. m. Rachabattuni, "Capacitor banks and its effects on the power system with high harmonic loads.," 04 2018.

[10] A. Pourramazan, S. Saffari, and A. Barghandan, "Study of failure mode and effect analysis (fmea) on capacitor bank used in distribution power systems," *IJIREEICE*, vol. 5, 01 2007.

[11] A. Emleh, A. Beer, H. Ferreira, and A. H. Vinck, "Interference detection on powerline communications channel when in-building wiring system acts as an antenna," in -, 09 2013.

[12] G. Traxler-Samek, R. Zickermann, and A. Schwery, "Cooling airflow, losses, and temperatures in large air-cooled synchronous machines," *Industrial Electronics, IEEE Transactions on*, vol. 57, pp. 172 – 180, 02 2010.

[13] M. Kutija and L. Pravica, "Effect of harmonics on ferroresonance in low voltage power factor correction system-a case study," *Applied Sciences*, vol. 11, p. 4322, 05 2021.

[14] T. Papadopoulos, I. Chaleplidis, A. Chrysochos, G. Papagiannis, and K. Pavlou, "An investigation of harmonic induced voltages on medium-voltage cable sheaths and nearby pipelines," *Electric Power Systems Research*, vol. 189, p. 106594, 2020.

[15] G. Cameron and P. Bodger, "Physical interpretation of inductive interference of power systems on communications circuits," *International Journal of Electrical Power and Energy Systems*, vol. 13, no. 3, pp. 153–159, 1991.

[16] X. Ou, K. Zhou, J. Zhang, X. Du, X. Tang, and X. Ou, "The quantitative analysis of electric meter measurement error with harmonic sources," in -, 01 2015.

[17] J. Pollefliet, "9 - ac-controllers," in *Power Electronics* (J. Pollefliet, ed.), pp. 9.1–9.18, Academic Press, 2018.

[18] M. Hossain, H. Pota, V. Ugrinovskii, and R. Ramos, "Excitation control for large disturbances in power systems with dynamic loads," in -, pp. 1 – 8, 08 2009.

[19] L. Abbassen and N. Benamrouche, "Dynamic behavior study of a synchronous generator connected to a grid," in -, 11 2023.

[20] J. Shen and Z. Zhu, "Sensorless flux-weakening control of permanent-magnet brushless machines using third harmonic back emf," *Industry Applications, IEEE Transactions on*, vol. 40, pp. 1629 – 1636, 12 2004.

[21] O. E. Gouda and A. Z. Dein, "Enhancement of the thermal analysis of harmonics impacts on low voltage underground power cables capacity," *Electric Power Systems Research*, vol. 204, p. 107719, 2022.

[22] N. Edomah, "Effects of voltage sags, swell and other disturbances on electrical equipment and their economic implications," *Electricity Distribution - Part 1, 2009. CIRED 2009. 20th International Conference and Exhibition (Proceedings)*, pp. 1 – 4, 06 2009.

[23] S. Bhattacharyya and S. Cobben, *Consequences of Poor Power Quality ï¿½ An Overview*, ch.-. -, 04 2011.

[24] D. Johnson, D. Johnson, K. Hassan, D. Johnson, and K. Hassan, "Issues of power quality in electrical systems," *International Journal of Energy and Power Engineering*, 01 2016.

[25] O. P. Mahela, A. G. Shaik, and N. Gupta, "A critical review of detection and classification of power quality events," *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 495–505, 2015.

[26] J. Shukla, B. K. Panigrahi, and P. K. Ray, "Power quality disturbances classification based on gramian angular summation field method and convolutional neural networks," *International Transactions on Electrical Energy Systems*, vol. 31, no. 12, p. e13222, 2021.

[27] A. K. Puliyadi Kubendran and A. K. Loganathan, "Detection and classification of complex power quality disturbances using s-transform amplitude matrix based decision tree for different noise levels," *International Transactions on Electrical Energy Systems*, vol. 27, no. 4, p. e2286, 2017. e2286 ETEP-15-0152.R3.

[28] E. G. Ribeiro, T. M. Mendes, G. L. Dias, E. R. Faria, F. M. Viana, B. H. Barbosa, and D. D. Ferreira, "Real-time system for automatic detection and classification of single and multiple power quality disturbances," *Measurement*, vol. 128, pp. 276–283, 2018.

[29] S. Wang and H. Chen, "A novel deep learning method for the classification of power quality disturbances using deep convolutional neural network," *Applied Energy*, vol. 235, pp. 1126–1140, 2019.

[30] J. M. J. Z. L. X. K. Chen and J. Wu, "Classification of power quality disturbances via deep learning," *IETE Technical Review*, vol. 34, no. 4, pp. 408–415, 2017.

[31] S. K. G. Manikonda, S. Gangwani, S. P. K. Sreckala, J. Santhosh, and D. N. Gaonkar, "Power quality event classification using convolutional neural networks on images," in *2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP)*, pp. 1–5, 2019.

[32] S. Mishra, C. N. Bhende, and B. K. Panigrahi, "Detection and classification of power quality disturbances using s-transform and probabilistic neural network," *IEEE Transactions on Power Delivery*, vol. 23, no. 1, pp. 280–287, 2008.

[33] S. Choudhury and G. K. Sahoo, "A critical analysis of different power quality improvement techniques in microgrid," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 8, p. 100520, 2024.

[34] S. Khan, B. Singh, and P. Makhija, "A review on power quality problems and its improvement techniques," in -, pp. 1–7, 04 2017.

[35] T. S. Key, "Diagnosing power quality-related computer problems," *IEEE Transactions on Industry Applications*, vol. IA-15, no. 4, pp. 381–393, 1979.

[36] R. Sangepu and T. V. Muni, "Effect of power quality issues in power system and its mitigation by power electronics devices," -, vol. 28, pp.–, 01 2015.

[37] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[39] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

[40] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[41] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400 – 407, 1951.

[42] Y. LeCun, L. Bottou, G. Orr, and K. Müller, *Efficient backprop*, pp. 9–48. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2012. Copyright: Copyright 2021 Elsevier B.V., All rights reserved.

[43] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[44] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 07 2011.

[45] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[46] T. T. and G. Hinton, "(2012) lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural Networks for Machine Learning, 4, 26-31.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[48] T. Dozat, "Incorporating nesterov momentum into adam," in -, 2016.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, p. 84 90, May 2017.

[50] Zhou and Chellappa, "Computation of optical flow using a neural network," in *IEEE 1988 International Conference on Neural Networks*, pp. 71–78 vol.2, 1988.

[51] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[53] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[54] M. Ouhda, K. Elasnaoui, M. Ouanan, and B. Aksasse, *Content-Based Image Retrieval Using Convolutional Neural Networks*, pp. 463–476. -, 01 2019.

[55] S. Mol, "Fish species classification using deep learning and appearance-based feature extraction," *Journal of Electrical Systems*, vol. 20, pp. 2531–2546, 04 2024.

[56] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Feature transfer learning for face recognition with under-represented data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[57] A. J. Shepley, "Deep learning for face recognition: A critical analysis," *CoRR*, vol. abs/1907.12739, 2019.

[58] S. Yu and J. Ma, "Deep learning for geophysics: Current and future trends," -, 03 2021.

[59] J. Quinn, J. McEachen, M. Fullan, M. Gardner, and M. Drummy, *Dive Into Deep Learning: Tools for Engagement*. SAGE Publications, 2019.

[60] R. Igual, C. Medrano, F. J. Arcega, and G. Mantescu, "Integral mathematical model of power quality disturbances," in *2018 18th International Conference on Harmonics and Quality of Power (ICHQP)*, pp. 1–6, 2018.

[61] J. Torres, *Python deep learning: Introducción práctica con Keras y TensorFlow 2*. Alpha Editorial, 2020.

[62] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, "Pytorch," *Programming with TensorFlow: solution for edge computing applications*, pp. 87–104, 2021.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[64] M. G. J. Jimenez, "Pq event generator," 2024. `https://github.com/Micke1995/ROPEC-2024` [Accessed: (6/22/2024)].

[65] M. Juarez, A. Zamora-Mendez, J. Ortiz-Bejar, J. C. Silva, J. Cerda, and M. Paternina, "Pq-syda: Power quality synthetic disturbances dataset," *2024 IEEE PES Generation, Transmission and Distribution Latin America Conference and Industrial Exposition (GTDLA). Ixtapa, Mexico*, p. Accepted, 2024.

[66] O. I. C. Robles, "2 - disturbance dataset development for machine-learning-based power quality monitoring in distributed generation systems: a practical guide," in *Monitoring and Control of Electrical Power Systems Using Machine Learning Techniques* (E. Barocio Espejo, F. R. Segundo Sevilla, and P. Korba, eds.), pp. 27–50, Elsevier, 2023.

[67] Na, "Ieee recommended practice for monitoring electric power quality," *IEEE Std 1159-2019 (Revision of IEEE Std 1159-2009)*, pp. 1–98, 2019.

[68] S. U. Khan, M. Mynuddin, D. M. A. Ahad, M. I. Hossain, M. J. Islam, and M. F. Kabir, "A comparative analysis of deep learning models for power quality disturbance classification," in *2023 IEEE World AI IoT Congress (AIIoT)*, pp. 0317–0323, 2023.

[69] Rahul, K. Dagar, and M. Gangadharappa, "Boosted convolutional neural networks based hybrid approach for power quality disturbances analysis," in *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 1–6, 2023.

[70] D. H. Chiam, K. H. Lim, and K. H. Law, "Detection of power quality disturbances using wavelet-based convolutional transformer network," in *2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, pp. 150–154, 2022.

[71] B. Y. Husodo, K. Ramli, E. Ihsanto, and T. S. Gunawan, "Real-time power quality disturbance classification using convolutional neural networks," in *Recent Trends in Mechatronics Towards Industry 4.0* (A. F. Ab. Nasir, A. N. Ibrahim, I. Ishak,

N. Mat Yahya, M. A. Zakaria, and A. P. P. Abdul Majeed, eds.), (Singapore), pp. 715–724, Springer Singapore, 2022.

[72] W. Deng, D. Xu, Y. Xu, and M. Li, "Detection and classification of power quality disturbances using variational mode decomposition and convolutional neural networks," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1514–1518, 2021.

[73] W. Hong, Z. Liu, and X. Wu, "Power quality disturbance recognition based on wavelet transform and convolutional neural network," in *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 193–197, 2021.

[74] H. Xue, A. Chen, D. Zhang, and C. Zhang, "A novel deep convolution neural network and spectrogram based microgrid power quality disturbances classification method," in *2020 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 2303–2307, 2020.

[75] S. S. Berutu and Y.-C. Chen, "Power quality disturbances classification based on wavelet compression and deep convolutional neural network," in *2020 International Symposium on Computer, Consumer and Control (IS3C)*, pp. 327–330, 2020.

[76] M. A. Ahajjam, D. B. Licea, M. Ghogho, and A. Kobbane, "Electric power quality disturbances classification based on temporal-spectral images and deep convolutional neural networks," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1701–1706, 2020.

[77] M. A. Rodriguez, J. Felipe Sotomonte, J. Cifuentes, and M. Bueno-Lopez, "Power quality disturbance classification via deep convolutional auto-encoders and stacked lstm recurrent neural networks," in *2020 International Conference on Smart Energy Systems and Technologies (SEST)*, pp. 1–6, 2020.

[78] S. Basumallik, "Voltage quality time series classification using convolutional neural network," 2019.

[79] M. Mohammadi, M. Afrasiabi, S. Afrasiabi, and B. Parang, "Detection and classification of multiple power quality disturbances based on temporal deep learning," in *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*, pp. 1–5, 2019.

[80] W. L. Rodrigues Junior, F. A. Silva Borges, R. d. A. Lira Rabelo, B. V. A. de Lima, and J. E. Almeida de Alencar, "Classification of power quality disturbances using convolutional network and long short-term memory network," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, 2019.

[81] Y. Dong, X. Ding, H. He, W. Zhao, and J. Li, "Voltage sag classification based on multi-task parallel convolutional neural network," in *IECON 2022 48th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6, 2022.

[82] M. Bindi, C. I. Garcia, A. Luchetta, F. Grasso, M. C. Piccirilli, L. Paolucci, and I. Aizenberg, "Classification of power quality disturbances using multi-valued neural networks and convolutional neural networks," in *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 01–08, 2022.

[83] X. Wenting, D. Chendong, W. Xuechun, and D. Jie, "Power quality disturbance identification method based on improved fully convolutional network," in *2022 5th Asia Conference on Energy and Electrical Engineering (ACEEE)*, pp. 1–6, 2022.

[84] H. G. R. Gouk and A. M. Blake, "Fast sliding window classification with convolutional neural networks," in *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*, IVCNZ '14, (New York, NY, USA), p. 114 118, Association for Computing Machinery, 2014.

[85] O. Florencias-Oliveros, “Real-life power quality sags,” tech. rep., University of CÃ¡diz, 2017.

Miguel Gabriel Juárez Jiménez

Aprendizaje por transferencia y sintonización fina para la identificación y clasificación de eventos

Universidad Michoacana de San Nicolás de Hidalgo

Detalles del documento

Identificador de la entrega
trn:oid:::3117:402940114

Fecha de entrega
7 nov 2024, 10:55 a.m. GMT-6

Fecha de descarga
7 nov 2024, 11:03 a.m. GMT-6

Nombre de archivo
Aprendizaje por transferencia y sintonización fina para la identificación y clasificación de evento....pdf

Tamaño de archivo
2.8 MB

138 Páginas

39,776 Palabras

147,017 Caracteres

12% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Fuentes principales

11% Fuentes de Internet

6% Publicaciones

0% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

Caracteres reemplazados
58 caracteres sospechosos en N.º de páginas
Las letras son intercambiadas por caracteres similares de otro alfabeto.

Texto oculto
369 caracteres sospechosos en N.º de páginas
El texto es alterado para mezclarse con el fondo blanco del documento.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Formato de Declaración de Originalidad y Uso de Inteligencia Artificial

Coordinación General de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo

A quien corresponda,

Por este medio, quien abajo firma, bajo protesta de decir verdad, declara lo siguiente:

- Que presenta para revisión de originalidad el manuscrito cuyos detalles se especifican abajo.
- Que todas las fuentes consultadas para la elaboración del manuscrito están debidamente identificadas dentro del cuerpo del texto, e incluidas en la lista de referencias.
- Que, en caso de haber usado un sistema de inteligencia artificial, en cualquier etapa del desarrollo de su trabajo, lo ha especificado en la tabla que se encuentra en este documento.
- Que conoce la normativa de la Universidad Michoacana de San Nicolás de Hidalgo, en particular los Incisos IX y XII del artículo 85, y los artículos 88 y 101 del Estatuto Universitario de la UMSNH, además del transitorio tercero del Reglamento General para los Estudios de Posgrado de la UMSNH.

Datos del manuscrito que se presenta a revisión		
Programa educativo	Maestría en ciencias en ingeniería eléctrica.	
Título del trabajo	Aprendizaje por transferencia y sintonización fina para la identificación y clasificación de eventos en la calidad de energía	
	Nombre	**Correo electrónico**
Autor/es	Miguel Gabriel Juárez Jiménez	1614873c@umich.mx
Director	Jaime Cerda Jacobo	jaime.cerda@umich.mx
Codirector	Alejandro Zamora Méndez	alejandro.zamora@umich.mx
Coordinador del programa	Norberto García Barriga	norberto.garcia@umich.mx

Uso de Inteligencia Artificial		
Rubro	**Uso (sí/no)**	**Descripción**
Asistencia en la redacción	si	Para escribir algunas ecuaciones.

Formato de Declaración de Originalidad y Uso de Inteligencia Artificial

Coordinación General de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo

Uso de Inteligencia Artificial		
Rubro	**Uso (sí/no)**	**Descripción**
Traducción al español	si	Para traducir fragmentos de un artículo que se escribió en inglés
Traducción a otra lengua	si	Para traducir la introducción al inglés.
Revisión y corrección de estilo	no	
Análisis de datos	no	
Búsqueda y organización de información	no	
Formateo de las referencias bibliográficas	no	
Generación de contenido multimedia	no	
Otro	no	

Datos del solicitante	
Nombre y firma	Miguel Gabriel Juárez Jiménez.
Lugar y fecha	Morelia Michoacán a 30 de octubre de 2024.