



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Identificación automática de emisores radar en base al espectro
de la señal*

Grado en Ingeniería Mecánica

ALUMNO: Manuel Peláez Gómez

DIRECTORES: José Ignacio Valles Cancela
José María Núñez Ortuño

CURSO ACADÉMICO: 2025-2026

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Identificación automática de emisores radar en base al espectro
de la señal*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

El presente Trabajo Fin de Grado aborda la identificación automática de emisores radar mediante el uso de técnicas de inteligencia artificial, concretamente redes neuronales convolucionales (CNN). En el contexto actual de la guerra electrónica, donde los radares modernos presentan características que dificultan su detección, los métodos tradicionales basados en correlación resultan insuficientes. Por ello, se propone un enfoque basado en el análisis de representaciones tiempo-frecuencia de las señales.

La metodología incluye la generación de una base de datos sintética de 64.000 imágenes de señales radar con distintas modulaciones (LFM, NLFM, Barker, Frank, Costas, P4, DPSK y SFH), empleando herramientas de simulación en Matlab. Estas señales se transforman en espectrogramas mediante tres técnicas: STFT, distribución de Choi-Williams y STFT reasignado. Posteriormente, se diseñan, entrenan y optimizan diferentes arquitecturas CNN para evaluar su capacidad de clasificación.

Los resultados obtenidos muestran que las CNN son capaces de identificar patrones complejos en las señales, alcanzando altas tasas de precisión incluso en condiciones de bajo nivel señal-ruido. En conclusión, el uso de *Deep Learning* constituye una alternativa viable y prometedora para mejorar la identificación automática de emisores radar, con potencial aplicación en sistemas de guerra electrónica.

PALABRAS CLAVE

Radar, Guerra electrónica, Redes neuronales convolucionales (CNN), *Deep Learning*, Espectrograma

AGRADECIMIENTOS

A los profesores del Centro Universitario de la Defensa, José Ignacio Valles Cancela y José María Núñez Ortuño, por su apoyo constante a lo largo de la realización de este trabajo y, en especial, por su compromiso con el impulso de líneas de investigación orientadas a la modernización de nuestra Armada.

A mis padres, por ser un pilar esencial durante estos cinco años de formación y por su paciencia ante mis desvaríos y peroratas.

Y, por supuesto, a la camareta 201, sin cuya amistad la realización de este trabajo hubiera sido la mitad de entretenida.

Contenido

Índice de Figuras	3
Índice de Tablas.....	6
1 Introducción y objetivos	7
1.1 Motivación	7
1.2 Objetivos del trabajo	7
2 Estado del arte	9
2.1 Sistemas radar pulsados	9
2.1.1 Fundamentos y parámetros característicos	9
2.2 Compresión de pulsos	11
2.2.1 Modulación de Frecuencia Lineal (LFM o Chirp).....	11
2.2.2 Codificación de frecuencia: códigos Costas	11
2.2.3 Modulación de Frecuencia No Lineal (NLFM).....	11
2.2.4 Saltos de Frecuencia Secuenciales (SFH).....	12
2.2.5 Codificación de fase.....	12
2.3 Guerra electrónica	13
2.3.1 Evolución histórica	14
2.3.2 Detección, clasificación, identificación y localización de señales radar	15
2.4 Técnicas de inteligencia artificial.....	16
2.4.1 La inteligencia artificial y el Deep Learning	16
2.4.2 Redes neuronales artificiales (ANN)	17
2.4.3 Redes neuronales convolucionales (CNN)	18
2.4.4 Procesado de señales mediante IA.....	20
2.5 Análisis espectral de señales radar	20
2.5.1 Espectrograma clásico	21
2.5.2 Distribución de Choi-Williams (CWD).....	22
2.5.3 Espectrograma STFT reasignado.....	23
3 Desarrollo del TFG.....	26
3.1 Problema a resolver.....	26
3.2 Metodología de trabajo	26
3.2.1 Definición de la base de datos sintética	27
3.2.2 Selección de parámetros radar	27
3.2.3 Generación de la base de datos	28

3.2.4 Diseño de la red neuronal (CNN-A).....	39
3.3 Optimización de la red neuronal (CNN-B).....	43
3.4 <i>Dataset</i> en distribución de Choi-Williams.....	47
3.5 <i>Dataset</i> en espectrograma STFT reasignado.....	47
3.6 Corrección de errores de generación y reentrenamiento (CNN-C).....	48
4 Resultados / Validación / Prueba.....	50
4.1 CNN-A.....	50
4.2 CNN-B.....	52
4.2.1 STFT.....	52
4.2.2 CWD.....	55
4.2.3 STFT reasignado.....	57
4.2.4 Robustez frente a reducción de SNR del modelo CNN-B.....	59
4.3 CNN-C en STFT y CWD.....	63
4.3.1 STFT.....	63
4.3.2 CWD.....	65
4.4 Robustez frente a reducción de SNR de las CNN definitivas.....	68
4.5 Pruebas con dos pulsos por imagen.....	69
5 Conclusiones y líneas futuras.....	73
5.1 Valoración de resultados.....	73
5.2 Líneas futuras.....	74
5.2.1 Empleo de Transfer Learning.....	74
5.2.2 Optimización de las CNN diseñadas y pruebas en tiempo real.....	74
5.2.3 Empleo combinado de SNN o SVM junto con las CNN desarrolladas.....	74
5.2.4 Estudiar el efecto de la resolución de imagen.....	75
6 Bibliografía.....	76
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales.....	79
Anexo II: Reflexiones Éticas y Sociales.....	81
Anexo III: Siglas y Acrónimos.....	83
Anexo IV: Repositorio GitHub.....	84

ÍNDICE DE FIGURAS

Figura 2-1. Esquema básico de funcionamiento de un radar pulsado	10
Figura 2-2. Ecuación que define el código Frank.....	12
Figura 2-3. Progresión cuadrática continua para códigos P4	13
Figura 2-4. El misil Kh-47M2 Kinzhal alcanza mach 10 (fuente: <i>Wikimedia</i>).....	13
Figura 2-5. Radiogoniómetro en el National Museum of the Great Lakes (Fuente: <i>Wikimedia</i>)	14
Figura 2-6. Mástil integrado de la JS Mogami (fuente: <i>Wikimedia</i>).....	15
Figura 2-7. Funcionamiento esquemático IA (fuente: [11]).....	16
Figura 2-8. Jerarquía IA	17
Figura 2-9. Red Neuronal Artificial (fuente: <i>wikimedia</i>).....	18
Figura 2-10. Esquema de un perceptrón (fuente: <i>udemy.com</i>).....	18
Figura 2-11. Esquema básico CNN (fuente: <i>wikimedia</i>).....	19
Figura 2-12. Representación STFT de radar pulsado con modulación Costas.....	21
Figura 2-13. Caracterización matemática de la Transformada de Fourier	21
Figura 2-14. CWD de pulso pulsado con modulación Costas.....	22
Figura 2-15. Caracterización matemática de la Distribución de Choi-Williams.....	23
Figura 2-16. Representación STFT reasignado de radar pulsado con modulación Costas	24
Figura 2-17. Caracterización matemática del espectrograma STFT reasignado	24
Figura 3-1. Base de datos CIFAR10.....	27
Figura 3-2. Definición de variables del <i>script</i> de generación.....	29
Figura 3-3. Llamada a <i>scripts</i> por modulaciones	29
Figura 3-4. Bucle de ejecución y <i>display</i> de tiempo	30
Figura 3-5. Configuración de producción	31
Figura 3-6. Gestión de directorios.....	31
Figura 3-7. Bucle de generación.....	31
Figura 3-8. Recorte y centrado temporal.....	32
Figura 3-9. Generación de espectrograma.....	32
Figura 3-10. Formateo y guardado	33
Figura 3-11. Código del <i>display</i> informativo	33
Figura 3-12. Código generación de señal LFM.....	33
Figura 3-13. Generación señal NLFM.....	34
Figura 3-14. Código generación de señal Costas	35
Figura 3-15. Código generación de señal SFH.....	35
Figura 3-16. Código generación de señal Barker	36

Figura 3-17. Código generación señal Frank	36
Figura 3-18. Segundo bucle generación señal Frank.....	37
Figura 3-19. Código generación de modulación P4	38
Figura 3-20. Código generación de señal DPSK.....	39
Figura 3-21. Estructura CNN-A	40
Figura 3-22. Importación del <i>Dataset</i> comprimido.....	41
Figura 3-23. Definición de <i>dataloaders</i>	41
Figura 3-24. División de la base de datos y pre-procesamiento.....	42
Figura 3-25. Optimización de la CNN-A	42
Figura 3-26. Guardado de mejor modelo	43
Figura 3-27. Importación de librería y definición estructura CNN	44
Figura 3-28. Instancia del modelo neuronal	45
Figura 3-29. Tamaño del tensor de salida por capa y número de parámetros de entrenamiento	45
Figura 3-30. Carga y preparación de la base de datos.....	46
Figura 3-31. Mecanismo de <i>early stopping</i> y guardado de mejor modelo.....	46
Figura 3-32. Código de generación de señal en CWD	47
Figura 3-33. Código de generación de señal STFT reasignado.....	48
Figura 3-34. Modificación de la normalización de la imagen en STFT.....	48
Figura 3-35. Modificación de la adición de ruido	49
Figura 3-36. Modificación del mapa de color y <i>resize</i>	49
Figura 4-1. Curvas de aprendizaje CNN-A	50
Figura 4-2. Matriz de confusión CNN-A	51
Figura 4-3. Análisis Grad-CAM de la CNN-A	52
Figura 4-4. Curvas de aprendizaje CNN-B en STFT	53
Figura 4-5. Matriz de confusión de la CNN-B en STFT	54
Figura 4-6. Análisis Grad-CAM de la CNN-B en STFT.....	55
Figura 4-7. Curvas de aprendizaje CNN-B en CWD	55
Figura 4-8. Matriz de confusión de la CNN-B en CWD.....	56
Figura 4-9. Análisis Grad-CAM de la CNN-B en CWD.....	57
Figura 4-10. Curvas de aprendizaje CNN-B en STFT reasignado.....	57
Figura 4-11. Matriz de confusión de la CNN-B en STFT reasignado.....	58
Figura 4-12. Análisis Grad-CAM de la CNN-B en STFT reasignado	59
Figura 4-13. Resumen de resultados con bajo SNR.....	59
Figura 4-14. Matriz de confusión de las pruebas de bajo SNR en STFT	61
Figura 4-15. Análisis Grad-CAM de las imágenes de la prueba de bajo SNR	61
Figura 4-16. Distribución de intensidades de las imágenes STFT y CWD.....	62

Figura 4-17. Distribución de intensidades de las imágenes STFT reasignado.....	63
Figura 4-18. Curvas de aprendizaje CNN-C en STFT	64
Figura 4-19. Matriz de confusión CNN-C en STFT	65
Figura 4-20. Análisis Grad-CAM de CNN-C en STFT	65
Figura 4-21. Curvas de aprendizaje CNN-C en CWD	66
Figura 4-22. Matriz de confusión de la CNN-C en CWD.....	67
Figura 4-23. Análisis Grad-CAM de la CNN-C en CWD.....	67
Figura 4-24. Resumen de resultados pruebas SNR de las CNN definitivas.....	68
Figura 4-25. Matriz de confusión CNN-C en STFT a -10 dB.....	68
Figura 4-26. Matriz de confusión CNN-C en CWD a -6 dB.....	69
Figura 4-27. Resumen de resultados pruebas con 2 pulsos de las CNN definitivas	70
Figura 4-28. Análisis Grad-CAM de 2 pulsos en STFT.....	71
Figura 4-29. Análisis Grad-CAM de 2 pulsos en CWD.....	71
Figura 4-30. Análisis Grad-CAM de 2 pulsos en STFT reasignado	72

ÍNDICE DE TABLAS

Tabla 2-1. Resumen limitaciones espectrogramas clásicos en CNN	22
Tabla 2-2. Resumen limitaciones CWD en CNN.....	23
Tabla 2-3. Resumen limitaciones espectrograma reasignado en CNN	25
Tabla 3-1. Intervalos de generación	28
Tabla 4-1. <i>Accuracy</i> final de la CNN-A.....	51
Tabla 4-2. Resumen de resultados de la CNN-B.....	52
Tabla 4-3. <i>Accuracy</i> final de la CNN-B en STFT	53
Tabla 4-4. <i>Accuracy</i> final de la CNN-B en CWD	56
Tabla 4-5. <i>Accuracy</i> final de la CNN-B en STFT reasignado	58
Tabla 4-6. Resumen resultados robustez a bajo SNR.....	60
Tabla 4-7. Resumen de resultados CNN-C	63
Tabla 4-8. <i>Accuracy</i> final de la CNN-C en STFT	64
Tabla 4-9. <i>Accuracy</i> final CNN-C en CWD.....	66
Tabla 4-10. Relación PRF y tiempo de ventana de recorte	70

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Motivación

“Una Armada que esté a la vanguardia tecnológica tanto en la Fuerza como en el resto de la organización, a la altura de las principales marinas del mundo, y que adopte tecnologías innovadoras de forma ágil para cumplir con éxito su misión”

Almirante Jefe de Estado Mayor de la Armada, *Plan Armada 2050* (2024)

La evolución del entorno estratégico actual plantea desafíos críticos para la Armada. En este contexto, la tecnología y los avances científicos se han convertido en los pilares de la superioridad competitiva, poniendo de manifiesto la importancia de la calidad frente a la cantidad de medios disponibles. Como señala el Almirante Jefe de Estado Mayor de la Armada (AJEMA) en el Plan 2050 [1], la soberanía tecnológica, la libertad y autonomía son factores estratégicos clave, subrayando que *'el acceso a información y datos es cada vez más una ventaja competitiva para mantener la soberanía nacional y la libertad de acción en los diversos dominios'*.

En este escenario, una mayor automatización del análisis de señales en el dominio electromagnético resulta un campo tecnológico crucial e indispensable. El presente trabajo se alinea con este objetivo del AJEMA, al buscar potenciar la capacidad de innovación propia y reducir la dependencia tecnológica, contribuyendo así a la superioridad tecnológica y a la autonomía operativa que la Armada demanda en las próximas décadas.

1.2 Objetivos del trabajo

El propósito de este trabajo radica en evaluar la viabilidad conceptual del empleo e integración de redes neuronales convolucionales (CNN) en el proceso de identificación de emisiones radar. Al automatizar la caracterización de estas señales mediante aprendizaje profundo, se pretende establecer una base sólida que permita la posterior vinculación de las emisiones con sus posibles plataformas de origen.

Para ello, se ha dividido el trabajo en diferentes etapas:

- Análisis de las estructuras y funcionamiento actual de diferentes CNN en su empleo para visión artificial.
- Estudio de las posibilidades actuales de representación de señales, además de su idoneidad para adiestramiento de algoritmos de *machine learning*.

- Generación de una base de datos sintética de imágenes de diferentes modulaciones empleadas habitualmente en radar mediante la herramienta *Matlab*, con la que poder entrenar una red neuronal en la discriminación intrapulso de las modulaciones seleccionadas.
- Realización de diferentes pruebas sobre las CNN entrenadas que permitan determinar la viabilidad e idoneidad de este método.

2 ESTADO DEL ARTE

2.1 Sistemas radar pulsados

Los sistemas radar (*radio detection and ranging*) constituyen uno de los pilares fundamentales de la vigilancia, detección e identificación de contactos en entornos navales. Su capacidad para detectar y caracterizar objetivos a larga distancia, independientemente de las condiciones meteorológicas o de iluminación, los convierte en elementos clave dentro de los sistemas de defensa y de conciencia situacional.

El radar no solo actúa como sensor, sino también como fuente de información, ya que cada emisor presenta unas características propias que lo diferencian y que pueden ser explotadas por sistemas de identificación.

2.1.1 Fundamentos y parámetros característicos

El funcionamiento conceptual del radar se basa en la emisión y recepción de ondas electromagnéticas. El transmisor genera pulsos que son radiados al espacio mediante una antena. Estas ondas se propagan a la velocidad de la luz hasta que inciden sobre un objeto, donde parte de la energía es reflejada en múltiples direcciones. Una fracción de esta energía regresa hacia la antena del radar y es procesada.

Dependiendo de la naturaleza de su señal y su modulación, los sistemas radar se clasifican en tres arquitecturas principales: radares pulsados, radares de onda continua (CW) y radares de onda continua modulada (FMCW).

El radar pulsado, mostrado en la Figura 2-1, es la arquitectura más extendida en el entorno militar, constituyendo la base de la mayoría de los sistemas de exploración aérea, así como de navegación [2]. Su funcionamiento consiste en la emisión de un tren repetitivo de pulsos de energía electromagnética de muy corta duración, seguidos de periodos de escucha para captar los ecos reflejados por los blancos. A diferencia de los radares de onda continua, los sistemas pulsados no son capaces de transmitir y recibir simultáneamente utilizando la misma antena [3]. Para gestionar este proceso, el sistema integra un dispositivo crítico denominado duplexer, que es el encargado de conmutar entre transmisor y receptor, lo que permite usar una misma antena, así como proteger al circuito receptor, que es más sensible.

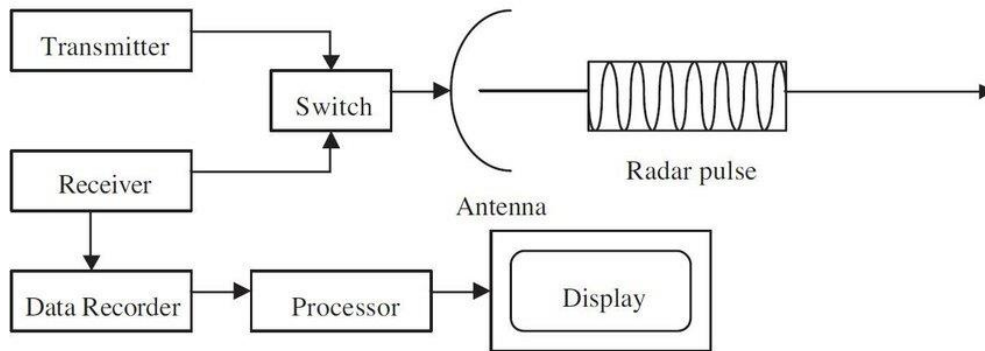


Figura 2-1. Esquema básico de funcionamiento de un radar pulsado

Durante la recepción, el sistema conmuta para aislar el transmisor y permitir que el receptor capte los ecos reflejados, los cuales presentan una potencia significativamente menor que la señal transmitida. El receptor amplifica y filtra estas señales para mejorar la relación señal-ruido. Un procesador mide el intervalo de tiempo entre la emisión del pulso y la recepción del eco. Dado que la velocidad de propagación es conocida, este dato permite calcular la distancia al blanco.

Los parámetros característicos de una señal de radar pulsador son los siguientes:

- Frecuencia portadora: determina la banda de operación. Las frecuencias bajas (bandas C a F según nomenclatura OTAN) ofrecen alcances mayores con menor precisión, utilizándose para vigilancia aérea a gran distancia. Las frecuencias altas (bandas I, J) proporcionan una mayor resolución, siendo más adecuadas para radares de navegación y direcciones de tiro [2].
- Ancho de banda (BW): es el rango de frecuencias que ocupa la señal.
- Intervalo de Repetición de Impulsos (PRI): Es el tiempo total que transcurre desde el inicio de un pulso hasta el inicio del siguiente. Incluye el tiempo de transmisión, el tiempo de recepción y un tiempo muerto o de recuperación [3].
- Frecuencia de Repetición de Impulsos (PRF): define el número de pulsos emitidos por segundo. La PRF es el factor que establece el alcance máximo inequívoco del radar; si un eco regresa después de que se ha iniciado la transmisión del siguiente pulso, el sistema calculará una distancia errónea.
- Duración o Ancho de Pulso (PW): representa el tiempo que el radar está transmitiendo activamente en cada ciclo. Un pulso ancho permite emitir más energía para alcanzar distancias mayores, pero tiene la contrapartida de una peor discriminación de contactos cercanos, ya que dos objetos dentro de la misma longitud de pulso aparecerán como un solo eco [3].
- Potencia: habitualmente expresada como Potencia Isotrópica Radiada Equivalente (PIRE), es el producto de la potencia del transmisor por la ganancia de la antena [3]. La potencia recibida disminuye drásticamente con la distancia, lo que exige un diseño equilibrado entre la sensibilidad del receptor y la potencia de emisión.

El análisis de estos parámetros constituye la base de las medidas de apoyo electrónico y serán presentadas en el apartado 2.3. Sin embargo, mientras que un radar trabaja en un rango de frecuencias y dirección concretas, los sistemas de guerra electrónica (EW) deben ser capaces de monitorizar un espectro amplio (desde señales de comunicaciones radio hasta los 20 GHz de equipos radar muy precisos) de forma permanente y omnidireccional [2].

En el ámbito de la inteligencia electrónica (ELINT), se explotan las pequeñas imperfecciones accidentales de los componentes del radar para generar la Identificación Específica del Emisor (SEI). Esta técnica actúa como una "huella dactilar" electrónica, permitiendo no solo identificar el modelo del radar, sino discriminar entre dos equipos idénticos de una misma serie, facilitando el levantamiento de un Orden de Batalla Electrónico (OBE) preciso del adversario [4].

Los radares pulsados modernos utilizan técnicas de modulación intra-pulso o compresión de pulsos, que permiten conseguir grandes alcances sin sacrificar resolución en distancia.

2.2 Compresión de pulsos

La compresión de pulsos, conocida igualmente como modulación intra-pulso, representa una metodología sofisticada en los sistemas de radar de impulsos para solventar la contradicción entre el alcance operativo y la precisión en la medición de distancias [5]. Bajo un esquema tradicional, la emisión de un pulso de gran duración facilita la propagación de una mayor carga energética, optimizando la detección de blancos remotos; no obstante, esta configuración compromete seriamente la capacidad de diferenciar objetivos que se encuentran en proximidad. Sin este recurso, los radares de largo alcance sufrirían de una baja resolución ante objetos cercanos, mientras que los de alta precisión verían drásticamente reducido su radio de acción.

A través de esta técnica, el sistema emite una señal de larga duración —y, por ende, de alta densidad energética— que, tras ser procesada en la etapa de recepción, se contrae en un pulso de tiempo reducido, optimizando así la resolución de distancia [5]. En el contexto de la guerra electrónica, esta capacidad resulta vital, dado que mejora la relación señal-interferencia y aumenta la resiliencia frente a los sistemas de perturbación del adversario [2]. A pesar de estas ventajas, es una propiedad técnica que también puede ser interceptada y analizada como una vulnerabilidad dentro del espectro electromagnético.

2.2.1 Modulación de Frecuencia Lineal (LFM o Chirp)

La técnica LFM (*Linear Frequency Modulation*), conocida popularmente como Chirp, se basa en la modificación lineal de la frecuencia de la portadora a lo largo de la extensión del pulso. Dicha transición puede ejecutarse de forma creciente (*up-chirp*) o decreciente (*down-chirp*). La ventaja técnica fundamental radica en que la resolución de distancia se independiza de la extensión temporal del impulso, pasando a ser inversamente proporcional al ancho de banda de la señal emitida. Gracias a este tipo de modulación, es factible obtener precisiones métricas o de centímetros utilizando pulsos con duraciones en la escala de los microsegundos [5].

2.2.2 Codificación de frecuencia: códigos Costas

Las secuencias o códigos Costas son una modalidad sofisticada de la técnica de saltos de frecuencia (*Frequency Hopping*) ejecutada a nivel intra-pulso. Este método consiste en fragmentar el impulso del radar en N sub-pulsos discretos, vinculando a cada segmento una frecuencia portadora diferenciada a partir de una secuencia previamente definido.

La configuración de estas frecuencias se determina mediante una matriz lógica, cuyo diseño asegura que la función de ambigüedad de la señal presente un lóbulo principal agudo y lóbulos laterales mínimos en el dominio tiempo-frecuencia [5]. En el ámbito de la Guerra Electrónica, la implementación de los códigos Costas resulta vital para las unidades navales por sus capacidades de Baja Probabilidad de Interceptación (LPI). Al dispersar la energía de la señal de forma pseudoaleatoria, se complica significativamente la detección y el análisis por parte de los sistemas de Guerra Electrónica del oponente, generando así ventajas tácticas y asimetrías favorables durante el enfrentamiento.

2.2.3 Modulación de Frecuencia No Lineal (NLFM)

A diferencia de la técnica LFM, la modulación NLFM (*Non-Linear Frequency Modulation*) varía la frecuencia de la portadora siguiendo una función no lineal (normalmente una función sigmoide o

tangente) durante la duración del pulso [6]. El propósito fundamental de esta técnica es realizar un control intrínseco sobre el espectro de la señal para reducir los lóbulos laterales en el proceso de compresión de pulso sin necesidad de aplicar funciones de ventana (*windowing*) adicionales. Desde el punto de vista de la ventaja competitiva en radares navales, la NLFM permite una mayor relación señal-ruido (SNR), ya que evita la pérdida de potencia asociada a los filtros de ponderación tradicionales. Esto se traduce en una capacidad de detección superior para blancos con sección radar (RCS) reducida en entornos de alta interferencia.

2.2.4 Saltos de Frecuencia Secuenciales (SFH)

La técnica SFH (*Sequential Frequency Hopping*) es una forma de saltos de frecuencia donde la portadora cambia de un valor a otro en una secuencia predeterminada o lineal mediante una serie de sub-pulsos. A diferencia de los códigos Costas, que buscan una matriz lógica para minimizar la ambigüedad, el SFH se utiliza principalmente para expandir el ancho de banda efectivo del sistema mediante la síntesis de frecuencias [9].

En el marco de la Guerra Electrónica, el SFH proporciona una excelente resistencia contra las contramedidas electrónicas (ECM, explicadas en el apartado 2.3) de banda estrecha. Al no concentrar la energía en una sola frecuencia, obliga al perturbador enemigo (*jammer*) a distribuir su potencia en un espectro más amplio, diluyendo su efectividad y protegiendo la capacidad de exploración del radar naval.

2.2.5 Codificación de fase

Las técnicas PMOP (*Phase Modulation on Pulse*) fragmentan un pulso largo en segmentos de igual duración denominados *chips*, asignando a cada uno una fase específica según un código predefinido.

2.2.5.1 Códigos binarios: Barker

Los códigos Barker emplean una modulación por desplazamiento de fase binaria (BPSK), donde la fase cambia entre 0° y 180°. *Los códigos Barker son únicos porque su función de autocorrelación produce lóbulos laterales con una amplitud máxima de 1/N respecto al lóbulo principal* [7]. Debido a esta propiedad, proporcionan un Nivel Máximo de Lóbulo Lateral (PSL) muy bajo, lo que evita detecciones falsas [7]. Sin embargo, su longitud está físicamente limitada (el código Barker más largo es de N=13). Para superar esto, los radares navales emplean códigos Barker anidados (por ejemplo, B5xB13), en los que se combinan los códigos para obtener secuencias de mayor longitud.

2.2.5.2 Códigos polifásicos: Frank

A diferencia de los códigos Barker o BPSK, que solo usan dos fases (0°, 180°), un código polifásico utiliza múltiples ángulos de fase (como 0°, 90°, 180°, 270°). Esto permite una transición más suave y generar una firma espectral más difícil de detectar para sistemas de Guerra Electrónica antiguos.

Los códigos Frank son una técnica polifásica que deriva de la discretización de una modulación LFM [9]. Es decir, se basa en una matriz de fases donde los saltos entre sub-pulsos siguen una progresión aritmética compleja, lo que imita la variación de frecuencia característica de las señales LFM.

Además, el cálculo de la fase de cada chip se basa en una expresión matemática (Figura 2-2) que relaciona el número de sub-pulso (n) con la columna o posición (k) y el valor del tamaño de matriz (M).

$$\phi_{\{n,k\}} = \frac{2\pi}{M}(n-1)(k-1)$$

Figura 2-2. Ecuación que define el código Frank

2.2.5.3 Modulación por Desplazamiento de Fase Diferencial (DPSK)

La técnica DPSK (*Differential Phase Shift Keying*) es una variante de la codificación de fase donde el cambio de fase se realiza respecto al segmento anterior y no a un valor absoluto.

En entornos navales, esta modulación es altamente valorada por su robustez frente a las variaciones de fase introducidas por el canal de propagación y el efecto Doppler. Al eliminar la necesidad de una referencia de fase coherente absoluta en el receptor para la demodulación inicial, los sistemas basados en DPSK simplifican el procesamiento de señal en condiciones climáticas adversas o estados de mar elevados, manteniendo la integridad de la identificación de la plataforma emisora [8].

2.2.5.4 Códigos Polifásicos P4

El código P4 pertenece a la familia de los códigos polifásicos de alta eficiencia y, al igual que el código Frank, deriva de la discretización de una modulación LFM. Sin embargo, el código P4 presenta una ventaja técnica crítica: es mucho más tolerante a los desplazamientos de frecuencia por efecto Doppler que los códigos Frank o Barker [9].

Para generar señales P4, se emplea una progresión cuadrática continua como la de la Figura 2-3.

$$\phi_k = \frac{\pi}{L}(k-1)^2 - \pi(k-1)$$

Figura 2-3. Progresión cuadrática continua para códigos P4

La resistencia Doppler de los códigos P4 permite que el radar mantenga una alta resolución de distancia incluso cuando rastrea plataformas que se desplazan a grandes velocidades. Esto la convierte en la modulación predilecta para su empleo en sistemas de defensa para misiles supersónicos como el de la Figura 2-4.



Figura 2-4. El misil Kh-47M2 Kinzhal alcanza mach 10 (fuente: Wikimedia)

2.3 Guerra electrónica

La Guerra Electrónica (GE), o *Electronic Warfare* (EW), se entiende como el conjunto de capacidades científicas y tácticas orientadas a garantizar el dominio del espectro electromagnético para las unidades propias, denegando simultáneamente su aprovechamiento al adversario. En el teatro de operaciones actual, la tecnología electrónica es un pilar tan determinante que, sin su concurso, las confrontaciones apenas se diferenciarían de los enfrentamientos navales ocurridos a finales del siglo XIX [2]. Tradicionalmente, la EW se estructura en tres pilares fundamentales:

- Medidas de Apoyo Electrónico (ESM): Orientadas a la detección, interceptación, clasificación y geolocalización de emisiones electromagnéticas con el fin de obtener una alerta temprana y soporte táctico inmediato.
- Contramedidas Electrónicas (ECM): Acciones de carácter ofensivo que emplean energía radiada para degradar o impedir el uso efectivo del espectro por parte del enemigo, recurriendo a la interferencia, el engaño o la neutralización.
- Medidas de Protección Electrónica (EPM): Estrategias diseñadas para blindar el uso propio de las radiofrecuencias frente a las contramedidas hostiles o las interferencias de carácter fortuito.

En el marco de las operaciones de apoyo, resulta vital diferenciar las ESM de la Inteligencia de Señales (SIGINT), la cual se subdivide en inteligencia de comunicaciones (COMINT) e inteligencia electrónica (ELINT). Mientras que las ESM poseen una finalidad táctica y de ejecución inmediata para lograr superioridad en el combate, la ELINT se especializa en la recolección de datos técnicos y el análisis de inteligencia a largo plazo sobre radares, sistemas de comunicación y emisores de interferencia. Esta labor de ELINT es la que permite confeccionar bibliotecas de amenazas detalladas y desarrollar contramedidas personalizadas para neutralizar sistemas de armamento específicos [4].

2.3.1 Evolución histórica

La Guerra Electrónica surgió como una disciplina técnica de forma prácticamente paralela a la implementación militar de las radiofrecuencias. A pesar de que las bases teóricas del electromagnetismo fueron establecidas por Maxwell en 1865 y validadas experimentalmente por Hertz años más tarde, el primer registro documentado de operaciones de EW no se produjo hasta la contienda Ruso-Japonesa (1904-1905) [2]. Durante este conflicto, se manifestaron dos hitos fundamentales que marcaron el inicio de esta rama táctica:

En primer lugar, la interceptación de señales de radio niponas por parte de la Armada Imperial Rusa, una acción que sirvió para alertar a sus fuerzas sobre una ofensiva inminente; este suceso se considera el bautismo del uso táctico de las ESM. El segundo acontecimiento relevante fue el nacimiento de las ECM, cuando un operador de radio ruso transmitió deliberadamente en la misma frecuencia que los equipos japoneses. Esta maniobra de interferencia logró bloquear las comunicaciones enemigas durante el asalto a Port Arthur, marcando un precedente en la denegación del espectro electromagnético [2].

Durante la Primera Guerra Mundial, se extendió el uso de la inteligencia de señales y aparecieron los primeros equipos de radiogoniometría (Figura 2-5). Su empleo fue clave en la victoria británica, que fue posible gracias a la interceptación de emisiones de los buques alemanes durante su salida a la mar.



Figura 2-5. Radiogoniómetro en el National Museum of the Great Lakes (Fuente: *Wikimedia*)

La Segunda Guerra Mundial supuso una revolución con la aparición del radar. Este avance tecnológico forzó el desarrollo de los primeros receptores de contramedidas, como el sistema alemán *Metox*, diseñado para detectar radares aliados, y el posterior uso de *chaff* (láminas metálicas) para enmascarar aeronaves [10]. También destacaron las operaciones de engaño electrónico masivo, como las llevadas a cabo antes del desembarco de Normandía, simulando los aliados una flota ficticia en Calais para confundir a los alemanes.

Los conflictos en el Medio Oriente en los años 70 demostraron la vulnerabilidad de las unidades navales ante misiles guiados, impulsando el desarrollo medio de EW más sofisticados.



Figura 2-6. Mástil integrado de la JS Mogami (fuente: *Wikimedia*)

En España, la industria nacional comenzó a potenciar el área de la EW en la década de los 80 con programas como el DENEb y el ALDEBARÁN, este último clave para dotar a las fragatas F-100 de un sistema integrado de EW [4]. Actualmente, la evolución se dirige hacia la digitalización total de los receptores (como demuestra el programa RIGEL) y la integración de sensores en mástiles cerrados para reducir la firma radar (Figura 2-6). Hoy en día, los buques de guerra han sustituido sus palos y velas por estructuras complejas que integran antenas y sensores destinados a asegurar el dominio del espectro electromagnético.

2.3.2 Detección, clasificación, identificación y localización de señales radar

El proceso de obtener conocimiento sobre el entorno electromagnético se fundamenta en una serie de tareas técnicas secuenciales ejecutadas por los sistemas ESM y ELINT:

- Detección: es el proceso inicial de intercepción de una señal mediante sensores especialmente dedicados. Estos receptores de banda ancha permiten al equipo de EW ser capaz de explorar el espectro a gran velocidad. La combinación de múltiples receptores colocados radialmente aporta información sobre la goniometría de la señal.
- Medida de parámetros y clasificación: una vez detectada la señal, se digitalizan y analizan los parámetros técnicos de la misma (descritos en el apartado 2.1.1).
- Identificación: es la fase final donde se discrimina si el contacto que ha generado la señal analizada es amigo, enemigo o neutral. Esto se logra mediante el cotejo automático o asistido de los parámetros analizados mediante el empleo de bases de datos generadas mediante captaciones ELINT previas [4].

- **Localización:** Se utiliza la goniometría para determinar el ángulo de llegada (AOA) de la señal. Mediante la triangulación desde múltiples plataformas o el uso de técnicas como la diferencia de tiempo de llegada (TDOA), se puede obtener la posición geográfica precisa del emisor [2].

2.4 Técnicas de inteligencia artificial

La Unión Europea (UE) define en su documento oficial [11] la inteligencia artificial (IA) como “...los sistemas que manifiestan un comportamiento inteligente, pues son capaces de analizar su entorno y pasar a la acción —con cierto grado de autonomía— ...”. Aunque también añade que no se trata únicamente de sistemas informáticos, sino que puede ir agregada en aparatos, como automóviles o electrodomésticos.

Estos modelos de inteligencia artificial funcionan al interactuar con su entorno a través de la captura y el análisis de información, ya sea organizada o dispersa. A partir de ahí, procesan el conocimiento obtenido para determinar las medidas más eficaces para alcanzar objetivos complejos (Figura 2-7).

En este ámbito, se distinguen dos vertientes metodológicas fundamentales: aquellas fundamentadas en procesos lógicos de razonamiento y las orientadas al aprendizaje automático. Lejos de ser una mera réplica del funcionamiento neuronal humano, la IA se trata de un área multidisciplinar que se nutre de diversos pilares de las matemáticas aplicadas, destacando especialmente el uso del álgebra lineal, el cálculo probabilístico, la teoría de la información y los métodos de optimización numérica [12].

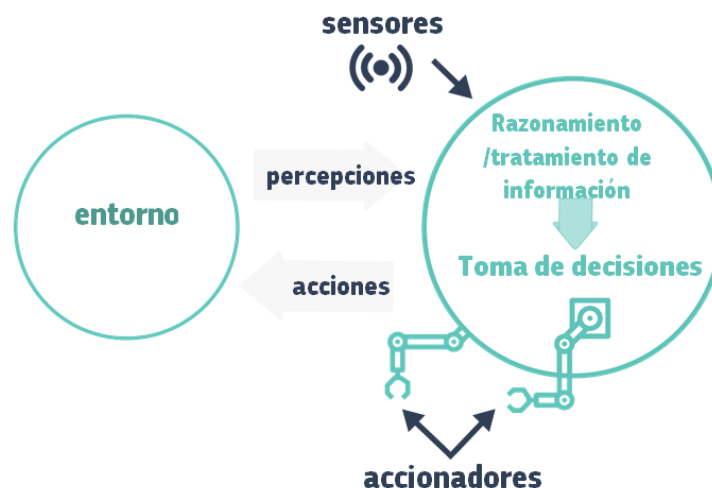


Figura 2-7. Funcionamiento esquemático IA (fuente: [11])

Históricamente, los sistemas de IA se enfrentaron primero a problemas intelectualmente difíciles para los humanos, pero sencillos para las computadoras [12]. Sin embargo, el verdadero reto en su desarrollo ha sido resolver tareas que los humanos realizan de forma intuitiva y automática, como el reconocimiento de voz o de imágenes. Para abordar estos problemas, la IA ha evolucionado desde sistemas basados en reglas hacia el aprendizaje automático (*Machine Learning*), permitiendo que las computadoras adquieran conocimiento a partir del análisis de datos del mundo real [12].

2.4.1 La inteligencia artificial y el Deep Learning

El aprendizaje profundo o *Deep Learning* es un tipo específico de aprendizaje automático que ha revolucionado la capacidad de los sistemas de IA para operar en entornos donde el ser humano aún sigue llevando la delantera. La relación entre las diferentes disciplinas de IA (Figura 2-8) se puede visualizar como una jerarquía donde la IA engloba al aprendizaje automático o *Machine Learning* y este al *Deep Learning*.

Mientras que el aprendizaje automático clásico depende en gran medida del diseño manual de características relevantes, el aprendizaje de representaciones permite que el algoritmo descubra por sí mismo no solo el mapeo de la entrada a la salida, sino también la propia representación de los datos [12]. Es decir, a diferencia de los algoritmos de Machine Learning, las redes neuronales de Deep Learning no necesitan de un trabajo previo de extracción de características, sino que es la propia red la que es capaz de extraerlas y ponderarlas automáticamente.

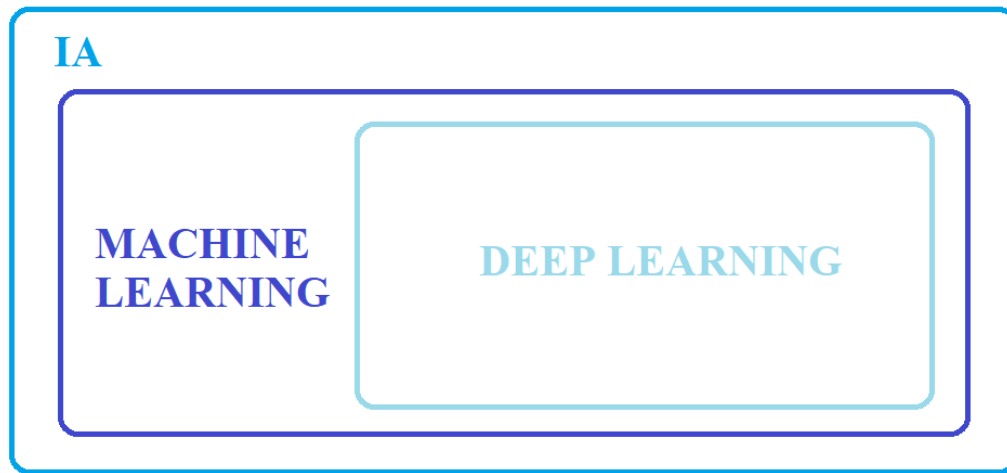


Figura 2-8. Jerarquía IA

Lo que permite al *Deep Learning* ser más eficiente en el aprendizaje de imágenes o representaciones es su capacidad para construir conceptos complejos basándose en otros de mayor simplicidad. Es decir, usa una serie de capas que permiten a cada una especializarse en tareas básicas y que se van superponiendo en el proceso de interpretación. Esta metodología de trabajo dota al *Deep Learning* de gran potencia y flexibilidad [12].

Por otro lado, cabe destacar que el éxito alcanzado por estas técnicas se debe principalmente al desarrollo de la informática, que permite procesar de forma más rápida y eficiente grandes conjuntos de datos.

2.4.2 Redes neuronales artificiales (ANN)

Las redes neuronales artificiales (ANN) son sistemas o modelos que se basan en la forma de funcionamiento del cerebro biológico, usando un sistema de unidades o nodos (neuronas) que al actuar en conjunto y comunicarse entre sí es capaz de dar respuestas complejas [13].

Una red neuronal (Figura 2-9) se compone de: una capa de entrada, una o más capas ocultas y una capa de salida. Cada capa está formada por una serie de nodos, conocidas como *perceptrons* (Figura 2-10), que son una imitación digital de la neurona biológica. Estos *perceptrons* son la unidad básica computacional de las redes neuronales y tienen un papel crucial en clasificación y reconocimiento de patrones.

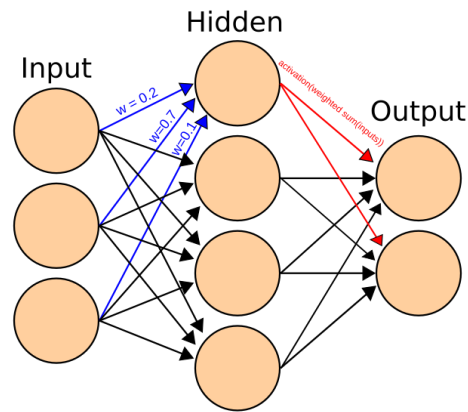


Figura 2-9. Red Neuronal Artificial (fuente: *wikimedia*)

Si las entendemos desde una perspectiva matemática, las capas que componen la red son funciones (entrada, activación u ocultas y salida) que reciben unos *inputs* y dan como respuesta unos valores de salida [13]. Existen redes en las que la información fluye unidireccionalmente (*feedforward*) y otras que emplean los valores de salida para retroalimentar la red. Este es el caso de las redes neuronales recurrentes (RNN), que gracias a esta característica poseen cierta memoria interna.

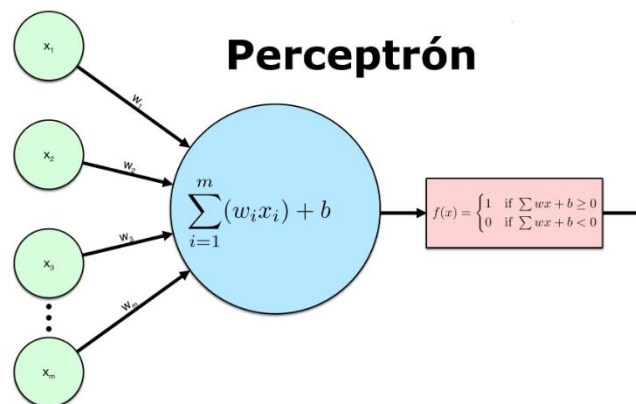


Figura 2-10. Esquema de un perceptrón (fuente: *udemy.com*)

2.4.3 Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (CNN) son una de las arquitecturas más utilizadas en *Deep Learning* y están compuestas por una estructura de nodos similar a las ANN.

Son un tipo de red *feedforward* diseñadas para procesar datos que poseen una topología conocida, como imágenes (cuadrículas 2D de píxeles) o series temporales [12]. Las CNN se distinguen por utilizar operaciones de convolución en una o varias capas para la extracción automática de características, en contraste con las redes neuronales tradicionales, que se basan principalmente en capas densas completamente conectadas.

El funcionamiento de las CNN se basa en tres ideas fundamentales que mejoran la eficiencia estadística y computacional [12]:

- *Sparse weight*: al usar filtros o *kernels* más pequeños que la entrada, la red tiene la capacidad de detectar características locales, como bordes o ángulos, usando menos parámetros y operaciones.
- *Equivarianza*: el mismo filtro se aplica a todas las posiciones de la entrada, lo que permite detectar la misma característica independientemente de su ubicación. Esto permite detectar objetos en una imagen aun cuando varían de posición o hacer reconocimientos de escritura con mayor porcentaje de acierto [14].

- Capas jerárquicas: mediante el empleo de múltiples capas especializadas, la red identifica porciones de la imagen cada vez mayores y más complejas. Es decir, mientras que las primeras capas se centran en detectar bordes o líneas, las capas finales son capaces de reconocer elementos en su conjunto.

La arquitectura básica de las CNN (Figura 2-11) consta de tres etapas principales: la etapa de convolución, la etapa de activación y la etapa de agrupación (*pooling*). El *pooling* sustituye la información de un elemento concreto a través de un análisis estadístico de los elementos circundantes (como píxeles), lo que permite que la red reconozca características cuando existe traslación de un elemento en la entrada.

Las funciones de activación introducen la no linealidad necesaria para que la red aprenda relaciones complejas. Tradicionalmente, se empleaban funciones como la sigmoide o la tangente hiperbólica, que provocaban un fenómeno llamado desvanecimiento del gradiente (*vanishing gradient* por su nombre en inglés), que estancaba la curva de aprendizaje cuando las redes de arquitectura profunda.

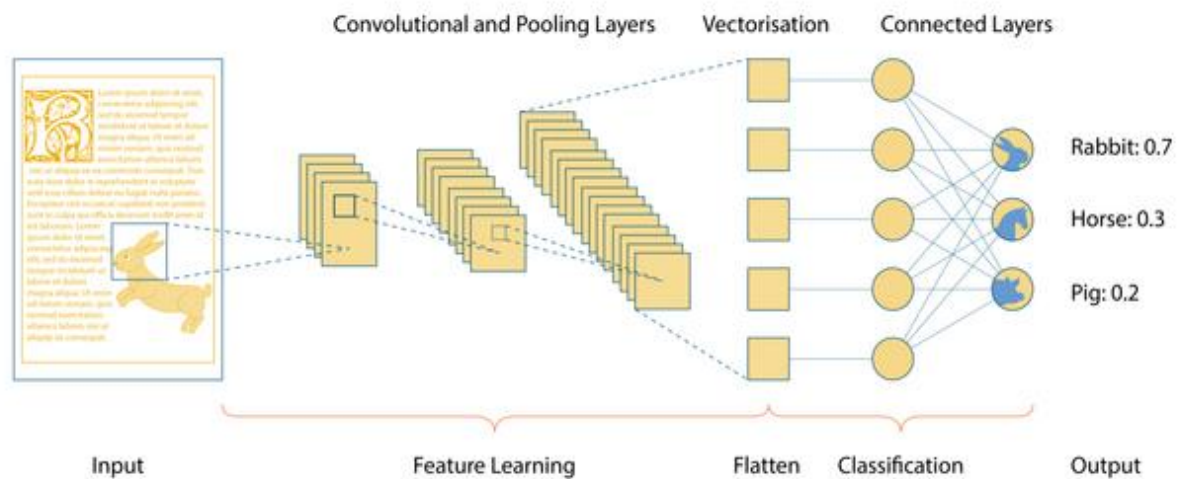


Figura 2-11. Esquema básico CNN (fuente: *wikimedia*)

Por otro lado, se emplean funciones lineales por tramos (de cero a infinito, ya que si la entrada es un número negativo la red lo toma como cero) como ReLu (Rectified Linear Unit), que permiten paliar también en problema del *vanishing gradient*. Además de su robustez matemática, la ReLu ofrece una razonable eficiencia computacional, lo que favorece que la red sea más selectiva y rápida al procesar la información visual.

Tras las capas de convolución y activación, se emplea una etapa de agrupación o *pooling*, cuya función principal es reducir la dimensionalidad espacial de los mapas de características. Este proceso disminuye el número de parámetros y el coste computacional del modelo, lo que ayuda a controlar el sobreajuste (*overfitting*). Además, el *pooling* otorga a la red una propiedad de invarianza local a la traslación. Esto permite que el sistema reconozca un patrón o rasgo específico, aunque su posición varíe.

Existen dos variantes predominantes en el diseño de CNN: el *Max Pooling* y el *Average Pooling*. El primero selecciona el valor máximo dentro de una ventana determinada lo que permite resaltar y conservar las características como bordes o texturas críticas. El *Average Pooling* calcula el promedio de los valores de la ventana, proporcionando una transición más suave y conservando una visión más general del contenido.

El proceso de aplanado (*flattening*) consiste en transformar las matrices multidimensionales resultantes de las etapas de convolución y *pooling* en un único vector unidimensional. Una vez aplanados, los datos ingresan en las capas *fully connected* o capas densas. En esta fase, cada neurona de una capa está vinculada con todas las neuronas de la capa anterior, permitiendo que el modelo combine las características detectadas (como formas, texturas y objetos complejos) para realizar una predicción.

Entre las arquitecturas clásicas de CNN destacan *LeNet-5*, *AlexNet*, *GoogLeNet* y *ResNet*. Estas redes han demostrado un rendimiento elevado en tareas de visión artificial, llegando incluso a igualar la capacidad humana en tareas relacionadas con el reconocimiento de objetos [12].

La evaluación del rendimiento de los modelos neuronales constituye un aspecto fundamental para garantizar su robustez y capacidad de generalización. En este contexto, herramientas como la función de pérdida, típicamente la *cross-entropy* en problemas de clasificación multiclase, cuantifica el grado de discrepancia entre las probabilidades predichas por la red y las etiquetas verdaderas, sirviendo como señal guía para el optimizador durante el proceso de aprendizaje de la red. Paralelamente, métricas como la precisión (*accuracy*), el *F1-score macro* (que asigna igual peso a cada clase independientemente de su prevalencia en el conjunto de datos) y la matriz de confusión permiten analizar no solo el desempeño global, sino también las confusiones específicas entre clases [12].

Adicionalmente, el monitoreo de parámetros de entrenamiento como las curvas de pérdida en conjuntos de entrenamiento y validación, junto con indicadores de complejidad computacional —tales como el número de parámetros, el tiempo de inferencia por muestra y el consumo de memoria—, proporciona una visión integral del equilibrio entre precisión y viabilidad práctica. Técnicas como el *early stopping* permiten optimizar el número de *epochs* necesarios para entrenar un modelo neuronal, reduciendo el coste computacional y haciendo más eficiente la optimización de la red.

2.4.4 Procesado de señales mediante IA

El reconocimiento de emisiones de radar es una tarea crítica para la explotación del espectro electromagnético, ya sea en entornos militares o civiles. Tradicionalmente, este proceso se basa en el análisis estadístico de parámetros como la frecuencia, el ancho de pulso (PW) y el intervalo de repetición de pulso (PRI), utilizando métodos de emparejamiento de parámetros o aprendizaje automático clásico [15]. No obstante, aunque pueda parecer que se han logrado avances importantes en el campo de la EW, los equipos ESM modernos siguen requiriendo de un operador adiestrado.

Por otro lado, el aumento de la complejidad en las modulaciones de los radares modernos y la necesidad de operar en un entorno electromagnético cada vez más ruidoso obligan a desarrollar métodos más capaces y eficientes. En [16] y [17] emplean CNN para afrontar estos problemas.

El procesado de señales mediante redes neuronales requiere del empleo de representaciones tiempo-frecuencia, como las que se presentan en el apartado 2.5, para alimentar bases de datos de adiestramiento y test, así como de técnicas avanzadas de IA o reducción del ruido. Es por ello por lo que han surgido las redes neuronales de picos (SNN) como alternativa a las ANN tradicionales [15].

Este es el motivo por el que la integración de redes neuronales convolucionales en la identificación automática de emisores radar es clave capturar patrones temporales complejos, así como características de compresión de pulsos que antes requerían estudios en profundidad, facilitando las labores de ESM y ELINT.

2.5 Análisis espectral de señales radar

El análisis de las señales de radar en el dominio del tiempo y la frecuencia es un pilar fundamental para la identificación automática de emisores. Debido a que las señales de radar modernas, especialmente aquellas de baja probabilidad de intercepción (LPI), presentan modulaciones complejas, la simple observación de la forma de onda en el tiempo resulta insuficiente [17]. Por ello, se recurre a representaciones bidimensionales que permiten extraer características intrínsecas del emisor, facilitando

el uso de técnicas avanzadas como el *Deep Learning*. A continuación, se detallan los tres tipos de espectrogramas más utilizados en el campo de las redes neuronales, analizando su naturaleza técnica y su impacto en el entrenamiento de redes neuronales.

2.5.1 Espectrograma clásico

El espectrograma clásico (Figura 2-12) es la herramienta tradicional de análisis tiempo-frecuencia, cuya filosofía se ha mantenido prácticamente inalterada desde la Segunda Guerra Mundial [18]. Técnicamente, se basa en la Transformada de Fourier de Tiempo Reducido (STFT), la cual divide una señal larga en segmentos más cortos mediante una función de ventana, aplicando la Transformada de Fourier a cada segmento para observar cómo cambia el contenido de frecuencia con el tiempo [19].

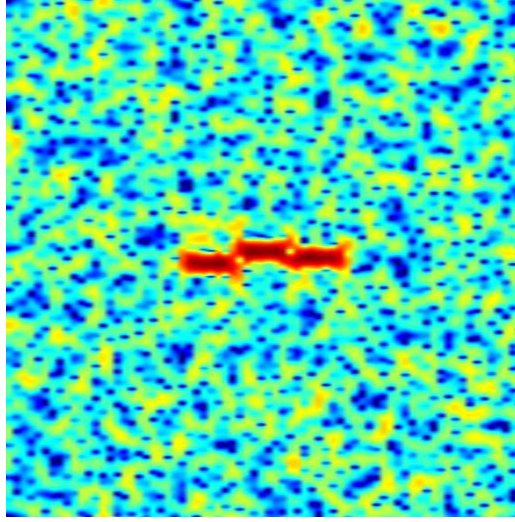


Figura 2-12. Representación STFT de radar pulsado con modulación Costas

La expresión de la Figura 2-13 describe el espectrograma de una señal discreta mediante la Transformada de Fourier. Para cada instante n y frecuencia k , se toma la señal $x[m]$ y se multiplica por una ventana $w[m - n]$, que selecciona un segmento local en el tiempo. Luego, ese segmento se proyecta sobre una exponencial compleja, que representa una frecuencia específica. La suma acumula estas contribuciones para obtener la transformada local. Finalmente, se calcula la magnitud al cuadrado, obteniendo la energía de cada frecuencia en ese instante, mostrando cómo evoluciona el contenido espectral en el tiempo.

$$S[n, k] = \left| \sum_{m=-\infty}^{\infty} x[m]w[m - n]e^{-j\frac{2\pi}{N}km} \right|^2$$

Figura 2-13. Caracterización matemática de la Transformada de Fourier

Este tipo de análisis espectral tiene ciertas limitaciones técnicas, ya que no es posible obtener una precisión infinita en los ejes temporal y frecuencial. Esto obliga a encontrar un compromiso entre ambos, lo que dificulta la determinación exacta del centro de la señal [18]. Esta difuminación en ciertas zonas de la señal se ve compensada con menores requerimientos computacionales al generarla.

Otra de sus desventajas es la falta de contraste en entornos de baja relación señal-ruido (SNR), en los que la señal de interés puede quedar enmascarada.

En la Tabla 2-1 se muestran resumidas las ventajas y limitaciones de este tipo de espectrogramas para su empleo como base de datos para CNN.

Ventajas	Limitaciones
Simplicidad	Falta de nitidez en bordes
Procesamiento rápido	Entorno bajo SNR

Tabla 2-1. Resumen limitaciones espectrogramas clásicos en CNN

2.5.2 Distribución de Choi-Williams (CWD)

La distribución de Choi-Williams (CWD), como la de la Figura 2-14, también llamada Distribución Exponencial (ED), representa uno de los avances más significativos en el análisis de señales y pertenece a la denominada clase de Cohen. Otras representaciones generan lo que se conoce como *cross-terms* u oscilaciones ficticias cuando se analizan señales con modulaciones complejas, siendo creada la CWD como una respuesta para el estudio de estas señales mediante estructuras de *Deep Learning* [20]. Su principal ventaja es la fidelidad al capturar la forma de la modulación de la señal.

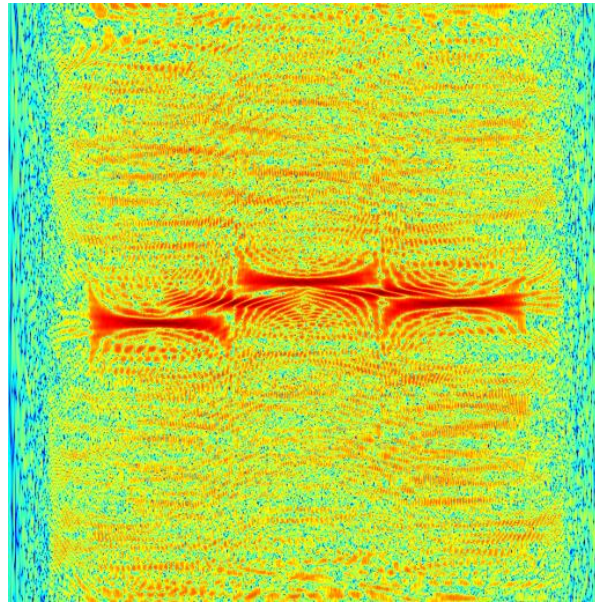


Figura 2-14. CWD de pulso pulsado con modulación Costas

La expresión matemática de esta representación (Figura 2-15) combina la señal $x[n]$ con su conjugado complejo desplazado, $x[n+p]x^*[n-p]$, lo que permite medir correlaciones locales alrededor del instante n . Estas correlaciones capturan cómo varía el contenido espectral en el tiempo.

Los índices m y p recorren desplazamientos en frecuencia y tiempo, respectivamente. La ventana $W_t[p]$ controla la extensión temporal del análisis, mientras que $W_f[m]$ actúa sobre la localización en frecuencia. Un elemento clave es el término exponencial con el parámetro σ , que introduce un decaimiento que atenúa las contribuciones lejanas en el plano tiempo-frecuencia, reduciendo así las interferencias cruzadas típicas de distribuciones cuadráticas.

La exponencial compleja $e^{-j2\pi km/N}$ realiza la proyección en frecuencia, permitiendo identificar qué componentes frecuenciales están presentes. La doble suma integra todas las contribuciones ponderadas, generando una representación que muestra cómo la energía de la señal se distribuye simultáneamente en tiempo y frecuencia, con un balance entre resolución y supresión de artefactos.

$$CWD[n, k] = 2 \sum_{m=-M}^M W_f[m] e^{-j \frac{2\pi}{N} km} \sum_{p=-P}^P W_t[p] \sqrt{\frac{\sigma}{4\pi m^2}} e^{-\frac{\sigma p^2}{4m^2}} x[n + p + m] x^*[n + p]$$

Figura 2-15. Caracterización matemática de la Distribución de Choi-Williams

La clave de la distribución CWD reside en el empleo de un filtro o *kernel* de tipo exponencial que actúa de mecanismo de control, limpiando la representación tiempo-frecuencia de interferencias. A su vez, el empleo del *kernel* preserva otras propiedades necesarias para el correcto análisis de la señal, como las variaciones de frecuencia o la energía de la señal [20].

La CWD es considerada una de las herramientas de pre-procesamiento más potentes para el entrenamiento de redes neuronales por varias razones. En primer lugar, la nitidez de la imagen, ya que, a diferencia de los espectrogramas clásicos, las imágenes generadas por CWD reflejan con gran precisión el periodo de modulación y el ancho de banda de señales complejas como LFM, códigos de Costas o códigos polifásicos (Frank, P1-P4) [17]. Esto proporciona una base adecuada para que las CNN extraigan patrones y características de la señal radar.

En segundo lugar, la CWD posee un rendimiento aceptable en escenarios de bajo SNR. Esto permite que modelos avanzados como *ResNet-152* o *Inception-v3*, como los descritos en [17], alcancen tasas de reconocimiento elevadas. En [17] se ha conseguido, mediante el empleo de redes neuronales entrenadas con bases de datos de imágenes CWD, una precisión del 97,8% en entornos con una SNR simulada de -2 dB.

Sin embargo, esta distribución también cuenta con algunas limitaciones. Una de ella es la existencia de imprecisiones visuales que pueden introducir sesgos en el proceso de entrenamiento de la CNN, sobre todo en entornos de ruido elevado. Otra desventaja es el mayor costo computacional, aunque se solventa con algoritmos basados en Transformada Rápida de Fourier (FFT).

En la Tabla 2-2 se muestran resumidas las ventajas y limitaciones de este tipo de espectrogramas para su empleo como base de datos para CNN.

Ventajas	Limitaciones
Nitidez imagen	Imprecisiones puntuales
Entorno bajo SNR	Costo computacional ¹

Tabla 2-2. Resumen limitaciones CWD en CNN

2.5.3 Espectrograma STFT reasignado

El espectrograma reasignado (Figura 2-16) es conocido por algunos autores como el "espectrograma del siglo XXI". Se trata de una evolución técnica que busca maximizar la precisión de la visualización espectral. Fue propuesto originalmente en la década de los 70, pero ha sido perfeccionada y ha ganado relevancia recientemente gracias a las mejoras en la capacidad de procesamiento y la posibilidad de empleo en áreas tan diversas como la ingeniería de materiales [21], medicina [22], fonética [18] o acústica submarina [23].

¹ Este problema se mitiga con algoritmos basados en FFT

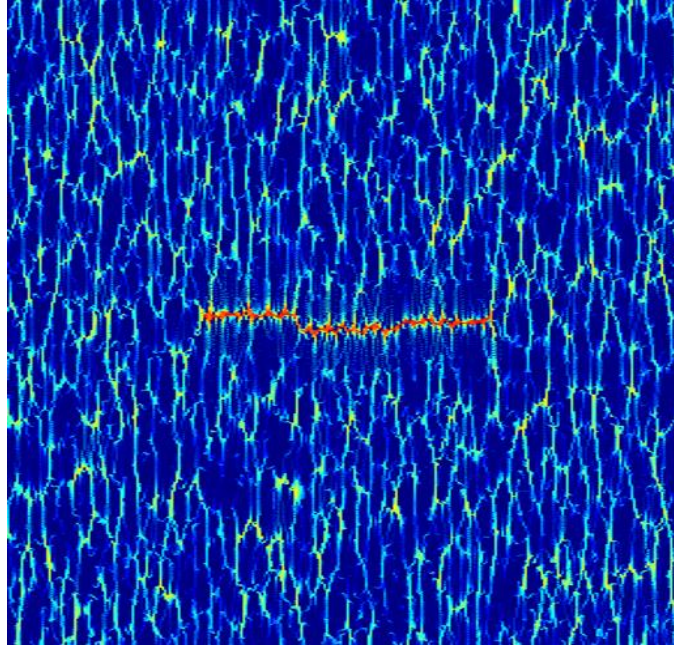


Figura 2-16. Representación STFT reasignado de radar pulsado con modulación Costas

La expresión matemática de esta representación (Figura 2-17) parte de $|X_w[m, p]|^2$, que es la energía obtenida de una transformada local (STFT). En lugar de mantener esa energía en la posición original, se redistribuye hacia nuevas coordenadas $(\hat{n}[m, p], \hat{k}[m, p])$, calculadas a partir de la fase. Las funciones delta ($\delta[n - \hat{n}]$ y $\delta[k - \hat{k}]$), que aseguran que la energía se reasigne exactamente a esos puntos. La doble suma acumula todas las contribuciones, produciendo un espectrograma más concentrado, con mejor resolución y menor dispersión en tiempo y frecuencia.

$$RS[n, k] = \sum_m \sum_p |X_w[m, p]|^2 \delta[n - \hat{n}[m, p]] \delta[k - \hat{k}[m, p]]$$

Figura 2-17. Caracterización matemática del espectrograma STFT reasignado

Este tipo de representación consiste fundamentalmente en mover o “reasignar” la energía. En lugar de asignar un valor de intensidad a cada punto, se identifican los “centros de gravedad” de la energía. De esta forma, las frecuencias o tiempos de menor intensidad se desplazan hacia los puntos de mayor concentración [24]. Por lo que, este tipo de representación es una mejora de la Transformada de Fourier de tiempo corto (STFT) que redistribuye los valores de energía calculados dentro de una ventana de análisis hacia el “centro de gravedad” real de cada componente espectral (tiempo-frecuencia) [24].

El resultado es la transformación de las zonas borrosas en líneas nítidas y estrechas, que dan una información más precisa de la frecuencia instantánea. Esto hace que los espectrogramas reasignados tengan una gran exactitud tanto en frecuencia como en tiempo, lo que facilitaría que las capas convolucionales de las CNN detecten las trazas características de la señal.

En [18] se afirma que “...el espectrograma reasignado parece, en principio, una herramienta eficaz para paliar algunos de los problemas de análisis que presentan los espectrogramas clásicos”. Además, mediante esta representación se obtiene una huella del emisor mucho más exacta, similar a la búsqueda que se realiza de la “huella vocal” en acústica forense para identificación de locutores.

Por otro lado, en señales de radar donde las frecuencias de modulación están muy cercanas entre sí, el espectrograma reasignado permite separar componentes que en el método clásico aparecerían fusionados en una sola mancha de energía [18], permitiendo a la red neuronal clasificar correctamente

emisores con parámetros técnicos similares. Esto podría eliminar la necesidad de usar dos redes neuronales (una CNN y una *Elman Neural Network* (ENN), que contiene una memoria interna que puede procesar texto) para distinguir con mayor precisión distintos tipos de señales en [16].

Sin embargo, entre las desventajas se encuentran que no supone una mejora notable en la obtención de parámetros respecto a los métodos tradicionales, sino que es una forma de examinar la relación tiempo-frecuencia que muestra únicamente las frecuencias instantáneas [24]. Además, la generación de este tipo de espectrogramas requiere aparentemente una mayor carga computacional, lo que dificultaría su empleo en grandes bases de datos.

En la Tabla 2-3 se muestran resumidas las ventajas y limitaciones de este tipo de espectrogramas para su empleo como base de datos para CNN.

Ventajas	Limitaciones
Gran nitidez y precisión	Relación coste computacional / mejora precisión
Separación elementos	Aplicaciones limitadas

Tabla 2-3. Resumen limitaciones espectrograma reasignado en CNN

3 DESARROLLO DEL TFG

3.1 Problema a resolver

El modelo tradicional de identificación en sistemas ESM se ha basado históricamente en la correlación de parámetros: el análisis del Intervalo de Repetición de Pulsos (PRI), el ancho de pulso (PW), la potencia de la señal, el tiempo de llegada (TOA) y la dirección de llegada (DOA) principalmente [25]. Sin embargo, en el entorno naval actual, esta manera de abordar el problema de la identificación se ve limitado con la proliferación de radares con agilidad de parámetros y el solapamiento de las librerías de inteligencia (distintos emisores comparten características casi idénticas). Este hecho, genera una ambigüedad en el proceso de discriminación que los algoritmos de comparación clásicos no pueden resolver con fiabilidad.

Para superar estas limitaciones, se propone el empleo de mecanismos de *Deep Learning* para llevar a cabo un análisis más complejo y preciso. Mediante el uso de Redes Neuronales Convolucionales se puede hacer el análisis de la señal directamente en representaciones tiempo-frecuencia, por lo que el sistema no lee datos aislados, sino que, usando algoritmos de visión artificial, es capaz de detectar patrones y señales específicas empleando toda la información contenida en el espectrograma.

3.2 Metodología de trabajo

La metodología de trabajo propuesta se basa en una serie de fases:

- Estudio de características de los radares de exploración modernos para definir la base de datos.
- Generación de 3 bases de datos sintéticas (STFT, CWD, STFT reasignado).
- Programación y entrenamiento de varias CNN en Python.
- Análisis de resultados y optimización de los modelos CNN.
- Validación de capacidades del modelo CNN.

Durante el desarrollo del proyecto se han empleado herramientas de asistencia basadas en modelos de lenguaje de gran tamaño (LLM), entre ellas Gemini, como apoyo en tareas de programación y exploración de soluciones. Estas herramientas se han utilizado principalmente para la generación de ejemplos de código, sugerencias de implementación y prototipos preliminares durante el desarrollo del sistema.

Las propuestas obtenidas fueron posteriormente revisadas, adaptadas y validadas antes de su integración en el sistema final desarrollado.

3.2.1 Definición de la base de datos sintética

Existen numerosas bases de datos en fuentes abiertas con imágenes para adiestrar algoritmos de visión artificial que reconozcan objetos o dígitos, como CIFAR10 (Figura 3-1) o ImageNet. Sin embargo, para el caso que se estudia en el presente trabajo, será necesario generar un *dataset* propio que reúna las características y parámetros típicos de funcionamiento de los radares modernos, al no existir bases de datos comerciales en este sentido.

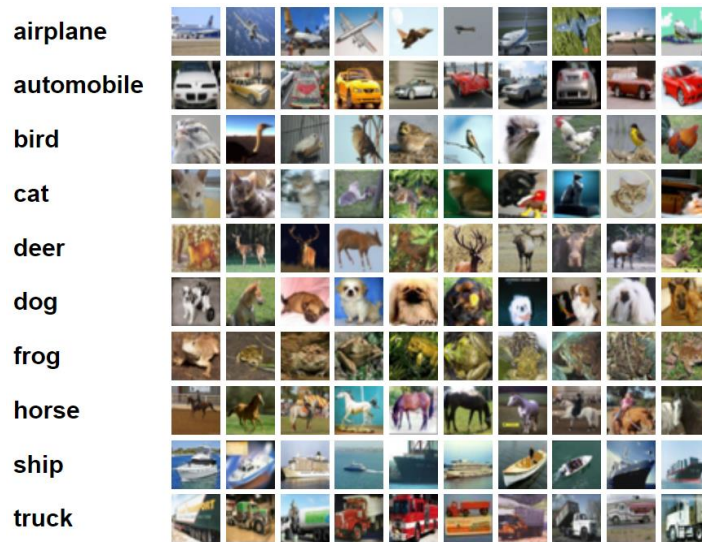


Figura 3-1. Base de datos CIFAR10

El diseño de radares actuales busca maximizar dos factores contrapuestos: el alcance (que requiere pulsos largos y mucha energía) y la resolución en distancia (que requiere de pulsos cortos y anchos de banda elevados). La compresión de pulsos (véase el apartado 2.2) permite resolver este problema, pero deja una “firma” en el pulso del radar que puede ser analizada para su clasificación.

La representatividad de la base de datos de entrenamiento es un factor crítico en el entrenamiento de un sistema de clasificación basado en *Deep Learning*, por lo que se ha seleccionado un conjunto de 8 modulaciones: LFM, NLFM, Barker, Frank, Costas, P4, DPSK, y SFH. Esta elección responde a un estudio de las características técnicas más extendidas en los radares de vigilancia y seguimiento de largo alcance en el ámbito naval y aeroespacial moderno [25].

Las modulaciones en frecuencia son el estándar en radares de búsqueda de largo alcance. La LFM es robusta y sencilla de implementar, mientras que la NLFM permite reducir los lóbulos laterales en el procesado, mejorando la discriminación en distancia cuando dos o más blancos se encuentran juntos.

La codificación de Fase (Barker y Frank) es una técnica que proporciona una alta resolución y es fundamentales en radares que requieren precisión en distancia. Los códigos Barker en concreto son un clásico por sus excelentes propiedades de autocorrelación, mientras que el código Frank es una aproximación escalonada de la LFM, lo que resulta en una mayor dificultad de interceptación. Por otro lado, los códigos polifásicos presentan estructuras de fase muy finas que permiten diferenciar radares de alta tecnología con características técnicas de LPI.

Además, se han elegido otro tipo de modulaciones como Costas, SFH o DPSK, cuyo uso está extendido en señales de comunicaciones (tanto civiles como militares), lo que permitirá que la red neuronal sea capaz de distinguirlas en el contexto radar y aportará robustez al modelo.

3.2.2 Selección de parámetros radar

Para poder generar las señales radar en el simulador de *Matlab* se deben seleccionar los parámetros que definen cada emisión. Para ello, se han obtenido de forma indirecta a partir de datos de diversos

radares de exploración aérea y de superficie. Se ha usado como fuente para obtener los parámetros [25], que cuenta con la mayor cantidad de información de radares en fuentes abiertas.

En la Tabla 3-1 se exponen los valores de los intervalos de generación que se usarán en los diferentes *scripts* de *Matlab*. Se ha seleccionado un ancho de pulso (PW) que variará entre 3 μ s y 20 μ s, un ancho de banda (BW) entre 1 MHz y 5 MHz y una relación señal a ruido (SNR) entre -6 dB y 8 dB (que es el estándar empleado en [17] para entornos electromagnéticos saturados).

Además, al centrarse la CNN en el estudio de las modulaciones intrapulso se generará un solo pulso y se fijará el PRF en 1000 Hz.

PW	BW	SNR	PRF
3-20 μ s	1-5 MHz	-6-8 dB	1000 Hz

Tabla 3-1. Intervalos de generación

Por otro lado, la frecuencia de muestro debe ser como mínimo el doble del ancho de banda, en este caso se ha fijado en 30 MHz, para tener resolución suficiente para hacer un estudio intrapulso de la señal; y el tamaño de ventana de “escucha” en 40 μ s, para dejar un margen de 10 μ s a cada lado de la imagen aun cuando la duración del pulso esté en su valor máximo.

3.2.3 Generación de la base de datos

El primer factor a tener en cuenta en el diseño de una base de datos es la cantidad de imágenes a generar. Aunque algunas fuentes afirman que, para conseguir que iguale el rendimiento humano, una CNN necesita un conjunto de 10 millones de imágenes etiquetadas para entrenarse, se pueden conseguir rendimientos elevados a partir de 5000 imágenes por categoría [12].

Con una cantidad de 8.000 imágenes por modulación y un total de 8 categorías, se generará una base de datos de 64.000 imágenes etiquetadas. De estas, se destinarán el 80% (51.200 imágenes) al entrenamiento, un 10% (6.400 imágenes) a validación y el 10% restante al test de la CNN.

Se ha seleccionado un tamaño de imagen de 518×518 (las arquitecturas de las redes neuronales que se han diseñado permiten que la resolución no sea múltiplo de dos) para el entrenamiento de la red neuronal, a fin de lograr un compromiso entre preservación de información espectral y eficiencia computacional, teniendo en cuenta las capacidades de hardware disponibles, en este caso una GPU (Graphics Processing Unit) NVIDIA RTX 4090. A pesar de que otras fuentes (véase [17]) consiguen emplear imágenes de 1024×104, el tamaño de imagen elegido permite mantener un nivel elevado de detalle en las representaciones tiempo-frecuencia, con el objetivo de que la red neuronal pueda identificar patrones finos asociados a las distintas modulaciones radar.

Además, la reducción de la resolución por debajo de 256×256 (como es el caso de las arquitecturas *ResNet* o *AlexNet*) podría provocar pérdida de información relevante, como variaciones sutiles en frecuencia o discontinuidades temporales. El tamaño elegido se mantiene dentro de un rango manejable desde el punto de vista computacional (la GPU empleada dispone de una capacidad de memoria de 24 GB de VRAM), lo que permite entrenar modelos con imágenes de este tamaño. Esto resulta especialmente relevante considerando el volumen total del *dataset* (64.000 imágenes), ya que la elección de un tamaño mayor (por ejemplo, superior a 768×768) incrementaría significativamente el consumo de memoria y por tanto el tiempo de entrenamiento.

El diseño del código se ha realizado en *Matlab* con la metodología siguiente:

- 8 *scripts* (1 por modulación) que generan la señal y las imágenes correspondientes.
- *Script* maestro en el que se selecciona la modulación y cantidad de imágenes a generar.

3.2.3.1 Diseño del Script maestro

Para el script maestro, se definen primero las variables a usar (Figura 3-2), que en este caso serán: el número de muestras o imágenes a generar, la frecuencia de muestreo, el número de pulsos, el tamaño de ventana (a partir del cual se calcula el número de muestras por ventana, y el directorio en el que se guardarán las imágenes. Además, se fija la FFT en 1024, ya que se recomienda que sea potencia de 2 para mejorar el rendimiento computacional. Este valor determina la “densidad” de información que va a tener la imagen generada, por lo que debe encontrarse un equilibrio entre “nfft” y resolución de imagen para no perder información ni rendimiento.

```
%% 1. CONFIGURACIÓN GLOBAL
cfg.n_muestras      = 100;
cfg.fs              = 30e6;
cfg.numpulses       = 1;
cfg.t_ventana_fija   = 40e-6;
cfg.num_muestras_win = round(cfg.t_ventana_fija * cfg.fs);
cfg.nfft            = 1024;
cfg.user_path        = 'C:\Users\mapeg\OneDrive\Documentos\TFG\1 DATASET\DataSet\STFT';
```

Figura 3-2. Definición de variables del *script* de generación

También se ha añadido un “selector” de las modulaciones a generar (Figura 3-3), con una llamada a los diferentes códigos generadores, lo que permite comentando líneas de código generar la base de datos por partes.

```
% 2. SELECCIÓN DE MODULACIONES
modulaciones = {
    'STFT_a_LFM', ...
    'STFT_b_COSTAS', ...
    'STFT_c_BARKER', ...
    'STFT_d_FRANK', ...
    'STFT_e_NLFM', ...
    'STFT_f_P4', ...
    'STFT_g_DPSK', ...
    'STFT_h_SFH' ...
};
```

Figura 3-3. Llamada a *scripts* por modulaciones

Se establecen también, como se ve en la Figura 3-4, un cronómetro para realizar una medición del tiempo de generación y el bucle con el que se ejecutan los códigos de las diferentes modulaciones. En caso de error en la ejecución de un *script*, se muestra en la ventana de comandos cuál ha sido el código que ha fallado, así como la causa del error.

```
% 3. EJECUCIÓN CÍCLICA
total_tic = tic;

for K = 1:length(modulaciones)

    % Extracción del nombre usando el nuevo índice
    script_actual = modulaciones{K};

    fprintf('\n>>> Ejecutando modulación %d de %d: %s\n', K, length(modulaciones),
script_actual);
    try
        % Ejecución
        eval(script_actual);
        fprintf('>>> %s finalizado con éxito.\n', script_actual);

    catch ME
        % en caso de fallo
        fprintf('\n[!] ERROR en el script: %s\n', modulaciones{K});
        fprintf('Causa: %s\n', ME.message);
    end
end

fprintf('\n=== PROCESO TERMINADO ===\n');
toc(total_tic);
```

Figura 3-4. Bucle de ejecución y *display* de tiempo

Finalmente, el código informa de que ha finalizado la generación y muestra el tiempo transcurrido por pantalla.

3.2.3.2 Diseño de los scripts por modulaciones

Para los 8 códigos que generan las señales radar se sigue la siguiente estructura:

1. Configuración de producción.
2. Gestión de directorios.
3. Bucle de generación masiva.
 - a. Selección aleatoria de parámetros.
 - b. Generación de la señal.
 - c. Recorte y centrado temporal.
 - d. Generación de espectrograma.
 - e. Formateo y guardado.
4. *Display* para mostrar información sobre el estado de ejecución del código.

De esta forma todos los códigos se pueden escribir a partir del mismo esqueleto, modificando únicamente la subsección B, en la que se genera la señal con la modulación específica.

Para poder llevar a cabo la generación de señales, se ha instalado la librería *Signal Processing Toolbox* de *Matlab*, que es oficial de *MathWorks*, que cuenta con funciones de alto nivel como “spectrogram()”, que calcula directamente la STFT.

En la sección 1 (Figura 3-5) se inicializan las variables, que llaman a las definidas en el *script* maestro, lo que facilita la generación del *dataset* ya que todo se controla desde un código principal. También se utiliza la función “tic” para iniciar un cronómetro interno que permita realizar una medición del tiempo de generación de las imágenes solicitadas. Por último, se establecen los intervalos de generación que se han definido en el apartado 3.2.2.

```
%% 1. CONFIGURACIÓN DE PRODUCCIÓN
n_muestras      = cfg.n_muestras;
fs              = cfg.fs;
num_muestras_win = cfg.num_muestras_win;
nfft            = cfg.nfft;
% INICIO DEL CRONÓMETRO GLOBAL
total_tic = tic;

% INTERVALOS DE GENERACIÓN
pw_range = [3e-6, 20e-6];
bw_range = [1e6, 5e6];
snr_range = [-6, 8];
```

Figura 3-5. Configuración de producción

En la sección 2 (Figura 3-6) se establece el directorio en el que se guardarán las imágenes generadas, que irán dentro de la carpeta destino, que se establece en la variable “folder_name”. En caso de que no exista la carpeta de destino, el propio *script* la creará.

```
%% 2. GESTIÓN DE DIRECTORIOS
user_path = cfg.user_path;
folder_name = ['LFM_STFT_', datestr(now, 'yymmdd_HHMM_'), num2str(n_muestras), ...
'pictures'];
output_dir = fullfile(user_path, folder_name);
if ~exist(output_dir, 'dir'), mkdir(output_dir); end
```

Figura 3-6. Gestión de directorios

La sección 3 () está dividida en subsecciones, de las cuales la única que varía es la subsección D. En primer lugar, se define el bucle de generación mediante la función “for”. Además, se inicia otro cronómetro (esta vez asociado a cada iteración) para medir el tiempo que tarda en generar cada imagen.

En la subsección A, se establece aleatoriamente el valor de PW, BW y SNR dentro del intervalo definido en la sección 1 (Figura 3-5) mediante la función “rand ()”. Los valores que se asignan varían en saltos decimales, es decir, de 1,4 se pasa a 1,5, 1,6 y así sucesivamente. De esta forma, se consigue la mayor representatividad en la generación de la base de datos y que la CNN sea capaz de generalizar.

```
%% 3. BUCLE DE GENERACIÓN MASIVA
for i = 1:n_muestras
    iter_tic = tic;

    % --- A. Selección aleatoria de parámetros ---
    pw = pw_range(1) + (pw_range(2) - pw_range(1)) * rand();
    bw = bw_range(1) + (bw_range(2) - bw_range(1)) * rand();
    snr = snr_range(1) + (snr_range(2) - snr_range(1)) * rand();
```

Figura 3-7. Bucle de generación

La subsección C es la encargada de ajustar una señal de cualquier longitud al tamaño de ventana establecido, asegurando que esta quede centrada en el dominio temporal. Para ello se usa una lógica en la que se plantean dos posibilidades: la señal es más grande que la ventana o la señal es más pequeña que la ventana. Se ha diseñado de esta manera para que las variaciones de tamaño de ventana o en el ancho de pulso máximo no afecten al código, evitando así errores en la ejecución de los *scripts*.

En la Figura 3-8 se puede ver cómo se ha programado esta lógica: primero se genera un “lienzo” en blanco mediante “sig_centered”. Si la señal es más grande que la ventana, se recorta el centro de la

misma y se coloca centrada; si la señal es más pequeña, se calculan el inicio y el final de la señal y se coloca dentro de ese intervalo, el resto estará vacío ya que se ha rellenado con ceros al inicio.

Por último, se añade ruido mediante la función “awgn” (Ruido Blanco Gaussiano Aditivo) a toda la señal generada según el valor de SNR que se ha establecido aleatoriamente. “Measured” se añade a la función de adición de ruido para que esta realice la medición de potencia de la señal contenida en el vector “sig_slice”, ya que, por defecto, la función asume que la potencia es 1 W, lo que generaría señales con una SNR real distinta de la establecida.

```
% --- C. RECORTE Y CENTRADO TEMPORAL ---
sig_centered = zeros(num_muestras_win, 1);
n_sig = length(sig);
if n_sig >= num_muestras_win
    start_idx = floor((n_sig - num_muestras_win)/2) + 1;
    sig_centered = sig(start_idx : start_idx + num_muestras_win - 1);
else
    start_target = floor((num_muestras_win - n_sig)/2) + 1;
    end_target = start_target + n_sig - 1;
    sig_centered(start_target:end_target) = sig;
end

% Ruido
sig_slice = awgn(sig_centered, snr, 'measured');
```

Figura 3-8. Recorte y centrado temporal

En la subsección D se transforma una señal unidimensional en una imagen, de naturaleza bidimensional, donde el eje horizontal representa la frecuencia y el eje vertical el tiempo transcurrido. El espectrograma analiza la señal en trozos, cuya longitud está definida por “window_len” (Figura 3-9). Esta ventana se va deslizando con un determinado solapamiento u “overlap” del 75%, por lo que la señal reutiliza parte de los datos anteriores para que la transición sea visualmente más suave. La función “spectrogram” da como resultado la matriz “P” que contiene la información en tiempo-frecuencia.

Para el procesamiento, se calcula el espectrograma de magnitud en escala logarítmica (decibelios), incorporando un factor de estabilización ϵ (“+ eps”) para prevenir singularidades matemáticas asociadas al logaritmo de cero. Se normaliza la imagen para que su punto más brillante sea siempre 0 dB y se establece el suelo de ruido en -40 dB para que simule la sensibilidad que tendría un hipotético sensor que recibe la señal.

```
% --- D. GENERACIÓN DE ESPECTROGRAMA CENTRADO EN FRECUENCIA ---

window_len = 32;
noverlap = floor(window_len * 0.75);
[~, ~, ~, P] = spectrogram(sig_slice, hamming(window_len), noverlap, nfft, ...
    fs, 'centered', 'yaxis');

% Procesamiento de magnitud
img_data = 10*log10(abs(P) + eps);
img_data = img_data - max(img_data(:));
img_data(img_data < -40) = -40;
```

Figura 3-9. Generación de espectrograma

Posteriormente, la subsección E (Figura 3-10) se encarga de transformar la matriz de datos numéricos del espectrograma en una imagen a color (RGB), estandarizar su tamaño y guardarla en el disco con un nombre descriptivo. Para ello, normaliza los valores de “img_data” (Figura 3-9) para que el rango final esté entre 0 y 1; convierte mediante la función “im2uint8()” los valores decimales a enteros de 8 bits y aplica el mapa de colores “jet” de *Matlab*. La función “ind2rgb()” convierte la matriz de una

capa en una imagen de 3 capas (rojo, verde y azul). Se redimensiona la imagen al tamaño establecido para la red neuronal y se guarda bajo el nombre que se define en la variable “nombre_file”.

```
% --- E. FORMATEO Y GUARDADO ---
img_norm = (img_data + 40) / 40;
img_rgb = ind2rgb(im2uint8(img_norm), jet(256));

% Redimensionado
img_final = imresize(img_rgb, [518 518]);

% Guardado
nombre_file = sprintf('LFM_STFT_BW%.1fM_PW%.1fu_SNR%d_%03d.png', bw/1e6,...
pw*1e6, round(snr), i);
imwrite(img_final, fullfile(output_dir, nombre_file));
```

Figura 3-10. Formateo y guardado

Finalmente, la sección 4 (Figura 3-11) establece un bucle para que se muestre por pantalla información sobre el estado de ejecución del código cada 5 iteraciones. En este caso, se imprime el número de imagen que se está procesando y el último archivo guardado. Finaliza el cronómetro “total_tic” que se inicializó al inicio del *script* y muestra en la ventana de comandos cuánto tiempo ha llevado la generación de las imágenes.

```
% --- 4. DISPLAY ---
if mod(i, 5) == 0 || i == 1
    clc;
    fprintf('Procesando: [%d/%d]\n', i, n_muestras);
    fprintf('Último archivo: %s\n', nombre_file);
end
end
t_final = toc(total_tic);
fprintf('\nGENERACIÓN FINALIZADA en %.2f segundos\n', t_final);
```

Figura 3-11. Código del *display* informativo

Esta estructura modular permite que las únicas variaciones entre los 8 códigos se produzcan en la subsección B, que es la encargada de la generación de la señal. Para generar señales con modulación LFM se ha diseñado el siguiente código:

Señales basadas en LFM

Este fragmento de código de MATLAB (Figura 3-12) utiliza la *Phased Array System Toolbox* para generar un pulso de radar con modulación LFM. Para ello emplea la función “phased.LinearFMWaveform()”, en la que se debe definir: frecuencia de muestreo, ancho de pulso, ancho de banda y el número de muestras para representar el pulso sin silencios. Además, “OutputFormat, Samples” sirve para indicar que la salida debe ser un vector de muestras de señal compleja.

Mediante “waveform()” se ejecuta el objeto como una función, se calcula la forma de onda analítica y se almacena en la variable “sig”. El resultado es un vector columna de números complejos que representa la fase y amplitud del pulso LFM.

```
% --- B. Generación de Señal---
waveform = phased.LinearFMWaveform('SampleRate', fs, ...
    'PulseWidth', pw, ...
    'SweepBandwidth', bw, ...
    'OutputFormat', 'Samples', ...
    'NumSamples', round(pw * fs));
sig = waveform();
```

Figura 3-12. Código generación de señal LFM

Señales basadas en NLFM

Para generar señales con modulación NLFM (Figura 3-13) se establece primero el número de muestras a partir del producto del ancho de pulso y la frecuencia de muestreo. Además, se crea el vector de tiempo “t” y se centra el tiempo en cero ya que el código usa una función tangente para definir la variación de frecuencia en el tiempo.

En segundo lugar, el código asegura que la variación de frecuencia coincida con el ancho de banda establecido y define la frecuencia instantánea. Se ha empleado “atan” para que la frecuencia cambie rápidamente en el centro del pulso y más lentamente en los extremos, lo que ayuda a reducir los lóbulos secundarios [6] y aporta realismo a la base de datos.

Por último, se calcula la fase integrando la frecuencia instantánea con “cumsum” (suma acumulativa) y se genera la señal compleja final empleando la fase calculada.

```
% --- B. Generación de señal NLFM ---
muestras_pulso = round(pw * fs);
t = (0:muestras_pulso-1)' / fs;
t_center = t - pw/2;

% función tangente o Sigmoidal
k = (bw / (2 * atan(5)));
inst_freq = k * atan(20 * t_center / pw);

% La fase es la integral de la frecuencia instantánea
phase = 2 * pi * cumsum(inst_freq) / fs;
sig = exp(1j * phase);
```

Figura 3-13. Generación señal NLFM

Señales basadas en códigos Costas

Para poder generar señales con modulación de fase Costas (Figura 3-14), se debe inicializar en la sección 1 (Figura 3-5) una variable que defina el intervalo de códigos Costas que se desea emplear. En este caso, se ha seleccionado un valor mínimo de 3 y máximo de 6, debido a que son los valores empleados en [17].

Como la señal se genera de forma manual, se define en primer lugar la duración de cada fase, que es el cociente del ancho de pulso y el valor de códigos Costas elegido. La construcción del pulso de la señal se va a realizar mediante el empleo de la función “randperm()”, que genera una permutación aleatoria de números del 1 al valor establecido de “sc” (códigos Costas). Por otro lado, se define el espaciado entre las frecuencias disponibles a partir del ancho de banda y se almacena en la variable “delta_f”. La última variable permite centrar las frecuencias en banda base.

Para la generación de la señal, se establece un bucle en el que se tienen en cuenta:

1. Número de muestras para cada salto de frecuencia.
2. Vector de tiempo “t_chip”, que crea el eje de tiempo para cada segmento.
3. Exponencial compleja “chip_f”. Esto permite saber si la frecuencia es positiva o negativa, lo que no se puede conseguir con funciones trigonométricas.

4. Concatenación de los segmentos generados al final de la señal acumulada y guardado en la variable “sig”.

```
% --- B. Generación de Señal Costas ---
pw_chip = pw / sc;
costas_pattern = randperm(sc);
delta_f = bw / sc;
freq_hops = (costas_pattern - (sc+1)/2) * delta_f;

sig = [];
for k = 1:sc
    n_muestras_chip = round(pw_chip * fs);
    t_chip = (0 : n_muestras_chip - 1)' / fs;
    chip_f = exp(1j * 2 * pi * freq_hops(k) * t_chip);
    sig = [sig; chip_f];
end
```

Figura 3-14. Código generación de señal Costas

Señales basadas en SFH

A diferencia de los códigos Costas, las señales con saltos de frecuencia generan estas variaciones de forma escalonada. Para simular este comportamiento en *Matlab* (Figura 3-15), se definen la duración de cada fase y su muestreo, así como se crea el vector de tiempo relativo, que empieza en 0 en cada salto de frecuencia. Por último, se generan los valores de frecuencia para cada escalón y se centran en banda base.

En bucle de generación define la frecuencia de cada “escalón” de la señal y genera la correspondiente señal compleja. Esta información se guarda en la variable “sig” en cada iteración, generándose así la señal radar.

```
% --- B. Generación de señal SFH ---
t_hop = pw / N; % Duración de cada escalón
n_muestras_hop = round(t_hop * fs);
t = (0:n_muestras_hop-1)' / fs;

sig = [];

% Escalera ascendente centrada en 0
freq_steps = (-(N-1)/2 : (N-1)/2) * df;

for k = 1:N
    f_actual = freq_steps(k);
    % Generamos el sub-pulso
    sub_pulso = exp(1i * 2 * pi * f_actual * t);
    sig = [sig; sub_pulso];
end
```

Figura 3-15. Código generación de señal SFH

Señales basadas en códigos Barker

Para poder generar señales Barker (Figura 3-16), se debe inicializar en la sección 1 (Figura 3-5) una variable que defina los códigos Barker que se van a emplear. Aunque en [18] se emplean los códigos Barker 7, 11 y 13, se ha optado por añadir además el valor 5, a fin de aportar la mayor representatividad posible a la base de datos.

"Para la generación manual de la modulación, se define la duración de cada elemento del código como el tiempo de chip (T_{chip}), calculado como el inverso del ancho de banda (definido como "bw" en la Figura 3-16). El número de muestras por chip se obtiene mediante el entero del producto de T_{chip} por la frecuencia de muestreo (fs), asegurando la coherencia en el dominio discreto."

En la parte de selección aleatoria de parámetros, se establece la variable "code_len", que elige aleatoriamente uno de los códigos Barker entre los 4 posibles. Se definen manualmente los códigos Barker y se establece la lógica condicional para que se seleccione automáticamente. Esta información se almacena en la variable "b_seq".

El bucle de generación crea un bloque de muestras con el valor de fase actual que almacena en la variable "chip". Por último, almacena la información con el mismo método de concatenación que se ha usado en modulación Costas (Figura 3-14).

```
% --- B. Generación de señal Barker ---
t_chip = 1 / bw;
n_muestras_chip = round(t_chip * fs);

% Obtención de secuencia Barker (0 y 180 grados -> +1 y -1)
switch code_len
    case 5, b_seq = [1 1 1 -1 1];
    case 7, b_seq = [1 1 1 -1 -1 1 -1];
    case 11, b_seq = [1 1 1 -1 -1 -1 1 -1 -1 1 -1];
    case 13, b_seq = [1 1 1 1 1 -1 -1 1 1 -1 1 -1 1];
end

sig = [];
for k = 1:code_len
    % Cada chip mantiene la misma fase
    chip = b_seq(k) * ones(n_muestras_chip, 1);
    sig = [sig; chip];
end
```

Figura 3-16. Código generación de señal Barker

Señales basadas en códigos Frank

Para la generación de una señal que emplea códigos Frank el código se encuentra en la Figura 3-17 y la), en primer lugar, se define el número de elementos de la secuencia Frank. Esta modulación se basa en una matriz de dimensiones $M \times M$, por lo que la longitud del código será el resultado de elevar "M" al cuadrado. En este caso, se le dan valores de 4, 5 y 6 en la sección de configuración de producción (Figura 3-5). El valor de la variable en cada iteración se obtiene de forma aleatoria con cada iteración.

La sección de generación de la señal está dividida en dos bucles:

1. Cálculo de la fase de cada chip (Figura 3-17).
2. Construcción de la señal compleja (Figura 3-18).

```
% --- B. Generación de señal Frank ---
t_chip = 1 / bw;
n_muestras_chip = round(t_chip * fs);
% Generación matriz de fase de Frank
frank_phases = zeros(1, L);
idx = 1;
for n = 1:M
    for k = 1:M
        frank_phases(idx) = (2 * pi / M) * (n-1) * (k-1);
        idx = idx + 1;
    end
end
```

Figura 3-17. Código generación señal Frank

En el primero, usa la expresión matemática de los códigos Frank (Figura 2-2) para realizar el cálculo de cada fase. El segundo bucle (Figura 3-18) almacena la fase en la variable “chip_phase” y después convierte el valor en un número complejo que representa un vector con un ángulo específico. Esta información se concatena en la variable “sig” para crear la señal.

```
% Construcción de la señal compleja (I/Q)
sig = [];
for k = 1:L
    % Generación chip con la fase correspondiente
    chip_phase = frank_phases(k);
    chip = exp(1i * chip_phase) * ones(n_muestras_chip, 1);
    sig = [sig; chip];
end
```

Figura 3-18. Segundo bucle generación señal Frank

Esta sección de código (Figura 3-17 y Figura 3-18) ha sido generada usando la herramienta de Inteligencia Artificial Generativa (AIG) “Gemini” en su versión “3 flash” [26], con el *prompt*: “genera con la misma estructura el código de generación Matlab de un pulso con modulación Frank. Su longitud será L y la frecuencia de muestreo fs”. La IAG había sido alimentada previamente con los códigos LFM, Costas y Barker ya generados, con el objetivo de que aprendiera la estructura que se estaba implementando hasta el momento y supiese que la información debe guardarse en la variable “sig”.

Señales basadas en códigos P4

Las señales P4 son otra variante de las secuencias polifásicas y se utiliza para simular una modulación lineal de frecuencia (LFM), aunque tienen mejores propiedades y son más robusta (2.2.5). Primero se define la variable “l_range” para establecer el intervalo de longitud del código, en este caso entre 16 y 64, que son tamaños estándar. En la subsección A (Figura 3-7), se calcula de forma aleatoria la longitud “L” a emplear en el bucle de generación.

Para generar la señal (Figura 3-19), se empieza definiendo la duración de cada “chip” y el número de muestras por cada uno de ellos, que es el entero del producto de la frecuencia de muestreo y “t_chip”.

Para crear las diferentes fases, se rellena con ceros un vector de longitud L y se inicia su bucle de generación. Las fases se calculan con una progresión cuadrática continua (Figura 2-3) y se guardan en “p4_phases(k)”.

La última sección construye la señal compleja mediante otro bucle, al igual que el código para NLFM, usando la función “exp()” y actualizando la información en la variable “sig”.

Esta sección de código (Figura 3-19) ha sido generada usando la herramienta de Inteligencia Artificial Generativa (AIG) “Gemini” en su versión “3 flash” [26], con el *prompt*: “usando la misma estructura, genera el código de Matlab para generar un pulso radar con modulación P4. El ancho de banda será bw y la frecuencia de muestreo fs”. La IAG había sido alimentada previamente con los códigos LFM, Costas, Barker, Frank y NLFM ya generados.

```
% --- B. Generación de señal P4 ---
t_chip = 1 / bw;
n_muestras_chip = round(t_chip * fs);

% Generación de fases P4
p4_phases = zeros(1, L);
for k = 1:L
    p4_phases(k) = (pi/L) * (k-1)^2 - pi*(k-1);
end

% Construcción de la señal compleja
sig = [];
for k = 1:L
    chip = exp(1i * p4_phases(k)) * ones(n_muestras_chip, 1);
    sig = [sig; chip];
end
```

Figura 3-19. Código generación de modulación P4

Señales basadas en modulación DPSK

Las señales DPSK no varían según una fase común, sino que se producen cambios de fase respecto al símbolo anterior. Para emular este comportamiento en *Matlab* (Figura 3-20), se definen el tiempo por fase, su número de muestras y se genera una secuencia aleatoria de 1 y 0 de longitud “L”.

Para la codificación diferencial, que es el núcleo de DPSK, se asigna espacio para las fases, con tamaño L+1, ya que se necesita una fase de referencia inicial; se inicializa la fase de referencia en 0 radianes y se guarda como primer elemento. El bucle de procesamiento de cada bit verifica si su valor es 1, realizando entonces su cambio de fase (“mod()”). Por último, se almacena la fase resultante para poder usarla como referencia en la siguiente iteración.

La última sección se encarga de construir la señal compleja. Para ello, emplea un bucle que recorre los vectores de fase creados para crear los fasores complejos correspondientes. La información resultante se va montando en la variable “sig”, al igual que se hace con los códigos de generación vistos hasta el momento.

Esta sección de código (Figura 3-20) ha sido generada usando la herramienta de AIG “Gemini” en su versión “3 flash” [26], con el *prompt*: “usando la misma estructura, genera el código de Matlab para generar un pulso radar con modulación DPSK. El ancho de banda será bw y la frecuencia de muestreo fs.”. La IAG había sido alimentada previamente con los códigos LFM, Costas, Barker, Frank, NLFM y P4 que ya habían sido diseñados.

```
% --- B. Generación de señal DPSK ---
t_chip = 1 / bw;
n_muestras_chip = round(t_chip * fs);

% Generación de bits aleatorios (0 y 1)
bits = randi([0 1], 1, L);

% Codificación diferencial
% d_phase(k) = d_phase(k-1) + pi * bit(k)
d_phases = zeros(1, L+1);
current_phase = 0;
d_phases(1) = current_phase;
for k = 1:L
    if bits(k) == 1
        current_phase = mod(current_phase + pi, 2*pi);
    end
    d_phases(k+1) = current_phase;
end

% Construcción de la señal compleja
sig = [];
for k = 1:length(d_phases)
    chip = exp(1i * d_phases(k)) * ones(n_muestras_chip, 1);
    sig = [sig; chip];
end
```

Figura 3-20. Código generación de señal DPSK

3.2.4 Diseño de la red neuronal (CNN-A)

El diseño de la red convolucional se realiza en Python y a través del entorno de programación *Jupyter Notebook*, que permite combinar código ejecutable, texto explicativo y visualizaciones en un mismo documento. La principal ventaja es que se puede escribir el código por celdas, lo que permite detectar y corregir errores de forma independiente, así como ver resultados de forma inmediata para optimizar el código de forma rápida.

En cuanto al diseño del código, en la primera celda (Figura 3-21), se importa la librería base de *PyTorch* para manejar tensores. Además, se importan los elementos constructivos de la red (como capas, funciones de pérdida...) y las funciones operacionales. Después se define la clase y se inicializa la clase padre (que es obligatorio en *Python* para el correcto funcionamiento de la red); además, se establece en 8 el número de clases a distinguir.

Se ha diseñado una arquitectura de red neuronal (Figura 3-21) inicial, que se ha llamado CNN-A, basada en una arquitectura de dos bloques convolucionales principales encargados del análisis jerárquico de la información visual de las imágenes de los espectrogramas. El primer bloque actúa como un extractor de rasgos primarios, utilizando 6 filtros para detectar componentes estructurales básicos. El segundo bloque incrementa la complejidad de la representación mediante 16 filtros, permitiendo la identificación de patrones morfológicos más específicos de cada modulación. En ambos casos, se emplean núcleos de convolución o *kernel* de 5 x 5 y funciones de activación *ReLU* para introducir las no linealidades esenciales, complementadas con capas de *Max Pooling* de 2 x 2 que reducen la dimensionalidad espacial y otorgan invarianza frente a pequeñas traslaciones. Se ha optado por *Max Pooling* y no por *Average Pooling* para evitar el suavizado de las imágenes, que podría generar pérdida de detalles y características críticas.

Se emplea también el ajuste dinámico de dimensiones (Figura 3-21). El código incluye un método especializado ("get_flatten_size") que calcula automáticamente el tamaño del tensor resultante tras las etapas de convolución y agrupación. Esto garantiza que la transición hacia las capas completamente

conectadas (*Fully Connected*) sea precisa, evitando errores de desajuste de datos al "aplanar" el mapa de características.

El bloque clasificador final se compone de tres capas lineales (Figura 3-21) que reducen progresivamente la información desde el espacio de características extraídas hasta un vector de 8 unidades, correspondiente al número de categorías de modulación a distinguir (LFM, NLFM, Costas, SFH, Barker, Frank, P4 y DPSK). La salida de la última capa es la encargada de generar los *logits* necesarios para la inferencia probabilística. Un *logit* se refiere a la salida escalar producida por la última capa lineal del modelo para cada clase. Estos valores constituyen puntuaciones que reflejan la evidencia relativa que el modelo asigna a cada clase, pero que no pueden interpretarse directamente como probabilidades, ya que no están acotados ni suman uno. Posteriormente, una función de activación final asigna a cada clase una probabilidad normalizada entre cero y uno, de forma que la suma de todas ellas es uno.

```
class CNNA(nn.Module):
    def __init__(self):
        super(CNNA, self).__init__()
        # Capa 1: Entrada 3 canales (RGB), 6 filtros, kernel 5x5
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool1 = nn.MaxPool2d(2, 2)

        # Capa 2: Entrada 6 canales, 16 filtros, kernel 5x5
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.pool2 = nn.MaxPool2d(2, 2)

        # --- AJUSTE DE DIMENSIONES DINÁMICO ---
        self.flatten_size = self._get_flatten_size((3, 518, 518))

        # Capas Fully Connected
        self.fc1 = nn.Linear(self.flatten_size, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 8)

    def _get_flatten_size(self, shape):
        x = torch.zeros(1, *shape)
        x = self.pool1(F.relu(self.conv1(x)))
        x = self.pool2(F.relu(self.conv2(x)))
        return x.numel()
```

Figura 3-21. Estructura CNN-A

Para reducir tiempo de subida de la base de datos en el entorno de programación, se suben las imágenes comprimidas (Figura 3-22), por lo que es necesario emplear “MultiRarImageFolder”. Esta clase permite leer imágenes directamente desde carpetas “.rar”. Por lo que cada una de estas carpetas comprimidas se interpreta automáticamente como una categoría o clase del modelo, facilitando una organización jerárquica y limpia del *dataset*. Durante la inicialización (“_init_”), el sistema realiza un escaneo de los archivos comprimidos para construir un índice mapeado (“simples”). Este índice registra la ubicación exacta de cada imagen dentro de su respectivo archivo RAR. Al mantener los “actuadores” de los archivos abiertos (“rar_handlers”), el sistema evita la sobrecarga que provocaría la de apertura y cierre constante de ficheros, acelerando el flujo de datos y paliando el gran coste computacional de esta operación.

Por otro lado, se emplea “_getitem_” para emplear una estrategia de *Lazy Loading*. Es decir, solo cuando el modelo solicita una imagen específica para un proceso de entrenamiento, la clase accede al archivo comprimido, extrae los bytes necesarios en memoria (“io.BytesIO”), los convierte a un formato de imagen procesable y aplica las transformaciones pertinentes.

```
# --- 1. DEFINICIÓN DEL DATASET PERSONALIZADO ---
class MultiRarImageFolder(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform
        self.samples = []

        rar_files = sorted([f for f in os.listdir(root_dir) if f.lower().endswith('.rar')])
        self.classes = [f.replace('.rar', '') for f in rar_files]
        self.class_to_idx = {cls_name: i for i, cls_name in enumerate(self.classes)}
        self.rar_handlers = {}

        for rar_name in rar_files:
            rar_path = os.path.join(self.root_dir, rar_name)
            label_idx = self.class_to_idx[rar_name.replace('.rar', '')]
            rf = rarfile.RarFile(rar_path)
            self.rar_handlers[rar_name] = rf

            for f in rf.infolist():
                if not f.is_dir() and f.filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                    self.samples.append((rar_name, f.filename, label_idx))

        print(f"Dataset cargado. Total de imágenes: {len(self.samples)}")
```

Figura 3-22. Importación del *Dataset* comprimido

A continuación, se establece también la división del *dataset* (Figura 3-24) según lo establecido en 3.2.3 y se aplica una semilla para permitir la reproducibilidad del proceso.

```
# --- 5. DATALOADERS OPTIMIZADOS PARA RTX 4090 ---
trainloader = DataLoader(
    train_dataset,
    batch_size=128,
    shuffle=True,
    num_workers=8,
    pin_memory=True
)
```

Figura 3-23. Definición de *dataloaders*

En la Figura 3-23 se puede ver la definición del “dataloader” correspondiente a la sección del *dataset* de entrenamiento. Los *dataloaders* son componentes encargados de gestionar la carga de datos para la red neuronal. En este caso, se establece el tamaño de lote en 64 empleando la variable “batch_size”. A su vez, el comando “shuffle=true” permite la aleatorización del orden en el que la red analiza las imágenes, evitando que esta memorice el orden. Por otro lado, se permite a la GPU acceder directamente a los datos de la memoria RAM mediante el comando “pin_memory=true”, lo que acelera la transferencia de datos a la red neuronal. “Num_workers=8” para permitir la paralelización de la carga de datos en la CPU.

El preprocesamiento (Figura 3-24) se aplica al conjunto de la base de datos mediante: “resize”, para asegurar que todas las imágenes tengan la misma dimensión; “ToTensor”, que convierte los píxeles a tensores (0,0 a 1,0); y “normalize”, que escala todos los valores a un rango entre -1 y 1. También se establece la división de la base de datos mediante el empleo de una *seed* o semilla, que permite la reproducibilidad garantizando que la secuencia de números pseudoaleatorios sea igual en cada ejecución. No se aplica ninguna medida de *Data Augmentation* en la base de datos de entrenamiento, ya que el empleo de técnicas de transformación ha aumentado notablemente la duración del entrenamiento de la red durante las pruebas.

```

# --- 3. CONFIGURACIÓN ---
data_dir = '../STFT_compressed'
full_dataset = MultiRarImageFolder(root_dir=data_dir)

# Proporciones: 80% Train, 10% Val, 10% Test
total_size = len(full_dataset)
train_size = int(0.80 * total_size)
val_size = int(0.10 * total_size)
test_size = total_size - train_size - val_size

# División física (con semilla para reproducibilidad)
train_base, val_base, test_base = random_split(
    full_dataset, [train_size, val_size, test_size],
    generator=torch.Generator().manual_seed(42)
)

# --- 4. TRANSFORMACIONES ---
train_transform = transforms.Compose([
    transforms.Resize((518, 518)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

```

Figura 3-24. División de la base de datos y pre-procesamiento

Para la optimización del modelo (Figura 3-25), se usan: “CrossEntropyLoss” para la función de pérdida; “Adam” para el optimizador, que ajusta la velocidad de aprendizaje para cada parámetro individualmente, y se fija la tasa de aprendizaje en 0,001; así como “optim.lr_scheduler.StepLR”, en el que se fija el “step_size” en 7 y “gamma” en 0,1. El primero le indica a la red que no haga cambios hasta haber realizado 7 *epochs*, mientras que el factor de reducción multiplica tras esos 7 *epochs* la tasa de aprendizaje por el valor de “gamma”. Este es un valor que sirve para variar el *learning rate* del optimizador, con el objetivo de reducirlo (en este caso multiplica el *learning rate* por 0,1) en caso de que el entrenamiento no mejore en 7 *epochs*.

```

# 2. Función de pérdida
criterion = nn.CrossEntropyLoss()

# 3. Algoritmo de Optimización
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 4. Scheduler
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)

```

Figura 3-25. Optimización de la CNN-A

El proceso de aprendizaje de la red neuronal se divide en 2 etapas:

1. Entrenamiento.
2. Validación.

De esta forma, es posible comprobar en cada *epoch* y de forma continua durante el aprendizaje si existe memorización u *overfitting*.

Por otro lado, se implementa un mecanismo de guardado del mejor modelo (Figura 3-26) que permite conservar automáticamente la versión óptima del modelo en caso de que el rendimiento comience a degradarse durante el entrenamiento. Este procedimiento emplea la función de pérdida de validación como criterio, almacenando exclusivamente los pesos correspondientes al *epoch* con el valor mínimo de dicha pérdida, evitando así el sobreajuste y garantizando la selección del punto óptimo de generalización del modelo.

```
# ===== GUARDAR EL MEJOR MODELO =====
if epoch_val_loss < best_val_loss:
    best_val_loss = epoch_val_loss
    torch.save(model.state_dict(), 'best_model_modulations.pth')
    save_msg = "(Mejor modelo guardado)"
else:
    save_msg = ""
```

Figura 3-26. Guardado de mejor modelo

En el proceso de diseño se han implementado una serie de celdas auxiliares que permitan comprobar la ejecución del proceso de entrenamiento y sus resultados de forma continua: visualización de una muestra de los datos de entrenamiento, de una muestra de imágenes donde se identifican las zonas de atención del modelo (*Grad-CAM*), información sobre las métricas de *loss* y *accuracy* en cada *epoch*, visualización de las gráficas de entrenamiento y la generación de la matriz de confusión.

El Grad-CAM es una técnica de inteligencia artificial utilizada para visualizar las decisiones tomadas por la red neuronal. Funciona generando mapas de calor que resaltan las regiones más importantes de una imagen que han influido en la predicción y toma de decisión de la red. Este análisis se hace empleando la última capa convolucional de la red, dando como resultado un mapa de activación que se superpone a la imagen original, permitiendo ver qué píxeles han activado más la red. Mediante este método, se puede comprobar si las predicciones de la red se basan en un análisis de parámetros irrelevantes o en la zona correcta de la imagen.

3.3 Optimización de la red neuronal (CNN-B)

Después del proceso de diseño inicial de red neuronal y tras analizar los resultados obtenidos por la CNN-A (véase el apartado 4.1), se ha abordado un proceso de optimización para lograr los siguientes objetivos:

1. Reducción del número de parámetros
2. Curva de aprendizaje más estable
3. Disminución del coste computacional
4. Mayor precisión en entornos con bajo SNR

La nueva arquitectura se ha denominado CNN-B y ha sufrido diversas modificaciones tanto en estructura como en la forma de ejecución.

Se ha modificado la arquitectura bajo el nombre CNN-B. El nuevo modelo está basado en cuatro bloques secuenciales con idéntica lógica (Figura 3-27), que dan una mayor profundidad y complejidad a la red. Esta disposición jerárquica permite que la extracción de características se realiza de forma más progresiva y, por tanto, el modelo tenga mejor capacidad de generalización: el Bloque 1 detecta bordes horizontales/verticales y manchas espectrales; el Bloque 2 identifica texturas y formas geométricas simples; el Bloque 3 es capaz de reconocer patrones complejos que conforman componentes distintivos de cada modulación radar; y el Bloque 4, el más abstracto, integra la disposición global y relaciones semánticas entre elementos, emulando el sistema visual humano. Junto a esta arquitectura se selecciona un tamaño de *batch* de 128 con el objeto de estabilizar las curvas de aprendizaje, ya que, al procesar un mayor número de muestras por iteración, el gradiente se estima de forma más precisa y las actualizaciones de los pesos son más estables.

Cada bloque genera un aumento progresivo del número de filtros (32, 48, 96 y 128), compensando así la reducción de resolución espacial inducida por las capas de *pooling* y enriqueciendo la representación en canales de características.

Para construir la distribución de capas de la red neuronal se usa “_make_layer”, que establece una misma estructura para todos los bloques: convolución, *batch* de normalización, activación *ReLU* y

agrupación, en este caso mediante *max pooling*, al igual que en la CNN-A. Esta estructura permite extraer características relevantes mientras se controla la dimensionalidad de las representaciones intermedias, favoreciendo un aprendizaje más eficiente. Además, el uso de normalización por lotes (*Batch Normalization*) tras cada convolución contribuye a estabilizar el entrenamiento al reducir la variación de las distribuciones de activación entre capas y la sensibilidad del modelo a cambios en los pesos, lo que permite el uso de tasas de aprendizaje más elevadas.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class CNN(nn.Module):
    def __init__(self, num_classes=8):
        super(CNN, self).__init__()

        # Bloque 1: Captura de bordes finos y ruido inicial
        self.layer1 = self._make_layer(3, 32) # Salida: 32 x 259 x 259
        # Bloque 2: Patrones de fase
        self.layer2 = self._make_layer(32, 64) # Salida: 64 x 129 x 129
        # Bloque 3: Estructuras complejas (saltos de frecuencia)
        self.layer3 = self._make_layer(64, 128) # Salida: 128 x 64 x 64
        # Bloque 4: Abstracción de alto nivel
        self.layer4 = self._make_layer(128, 256) # Salida: 256 x 32 x 32

        # Global Average Pooling
        self.gap = nn.AdaptiveAvgPool2d(1)

        # Clasificador final
        self.classifier = nn.Sequential(
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(128, num_classes)
        )

    def _make_layer(self, in_channels, out_channels):
        return nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
```

Figura 3-27. Importación de librería y definición estructura CNN

Después de definir la estructura, se emplea *global average pooling*, que actúa como traductor convirtiendo los mapas visuales en un resumen numérico sencillo. Su principal ventaja es la significativa reducción de parámetros con respecto al *flatten*. En este caso, la última capa tiene 256 filtros de 32 x 32 píxeles, por lo que, si se emplease *flatten*, la siguiente capa lineal tendría más de 260.000 valores. Sin embargo, al emplear la función “gap”, la red hace un promedio de cada matriz de 32 x 32, por lo que se queda con únicamente 256 valores, reduciendo así el *overfitting*.

Por otro lado, se crea el clasificador final, que será el encargado de interpretar la información extraída por cada bloque. Para ello, se usa una capa lineal para buscar correlaciones entre cada uno de los 256 rasgos extraídos previamente y “nn.ReLu” para eliminar las combinaciones que no aportan valor.

La principal diferencia con el diseño de la CNN-A, es la implementación del *dropout*, al que se le da un valor del 40%, lo que apaga aleatoriamente esa cantidad de neuronas durante el proceso de entrenamiento. El empleo de *dropout* permite que el modelo sea más robusto y tenga mejor capacidad de generalización, además de reducir el sobreajuste, al evitar que el modelo aprenda de memoria detalles.

Finalmente, una segunda capa lineal termina de reducir a 8 neuronas, igual que el número de categorías a estudiar. Cada neurona emite un valor *logit*, que permite que la red haga predicciones sobre la imagen.

La última parte de la celda (Figura 3-28) establece el camino a seguir por los datos desde que entran como imagen hasta que salen como una predicción mediante “def forward ()”. También se instancia el objeto de la red, que reserva los espacios de memoria para todos los pesos y crea la estructura que se ha definido previamente.

```
def forward(self, x):
    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.gap(x)
    x = torch.flatten(x, 1)

    return self.classifier(x)

model = CNN()
print(model)
```

Figura 3-28. Instancia del modelo neuronal

Antes de importar la base de datos, se usa la librería *torchsummary* para generar una tabla detallada (Figura 3-27) que muestra: tamaño del tensor de salida en cada capa y cuántos parámetros tiene que aprender la red. Esto permite llevar control de la red diseñada antes de comenzar con el entrenamiento.

Layer (type)	Output Shape	Param #
=====		
Conv2d-1	[-1, 32, 518, 518]	896
BatchNorm2d-2	[-1, 32, 518, 518]	64
ReLU-3	[-1, 32, 518, 518]	0
MaxPool2d-4	[-1, 32, 259, 259]	0
Conv2d-5	[-1, 48, 259, 259]	13,872
BatchNorm2d-6	[-1, 48, 259, 259]	96
ReLU-7	[-1, 48, 259, 259]	0
MaxPool2d-8	[-1, 48, 129, 129]	0
Conv2d-9	[-1, 96, 129, 129]	41,568
BatchNorm2d-10	[-1, 96, 129, 129]	192
ReLU-11	[-1, 96, 129, 129]	0
MaxPool2d-12	[-1, 96, 64, 64]	0
Conv2d-13	[-1, 128, 64, 64]	110,720
BatchNorm2d-14	[-1, 128, 64, 64]	256
ReLU-15	[-1, 128, 64, 64]	0
MaxPool2d-16	[-1, 128, 32, 32]	0
AdaptiveAvgPool2d-17	[-1, 128, 1, 1]	0
Linear-18	[-1, 64]	8,256
ReLU-19	[-1, 64]	0
Dropout-20	[-1, 64]	0
Linear-21	[-1, 8]	520
=====		
Total params: 176,440		
Trainable params: 176,440		
Non-trainable params: 0		

Figura 3-29. Tamaño del tensor de salida por capa y número de parámetros de entrenamiento

La segunda mejora aplicada a la red neuronal es la lectura de la base de datos directamente sin comprimir (Figura 3-30). De esta forma se elimina el cuello de botella que provoca leer de una en una las carpetas comprimidas, lo que previsiblemente reducirá el coste computacional.

```
# --- 1. CONFIGURACIÓN DE RUTAS Y TRANSFORMACIONES ---
data_dir = '../STFT' # Ruta a la carpeta donde descomprimiste las 8 modulaciones
|
train_transform = transforms.Compose([
    transforms.Resize((518, 518)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

test_transform = transforms.Compose([
    transforms.Resize((518, 518)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# --- 2. CARGA DEL DATASET (DIRECTO DESDE CARPETAS) ---
full_dataset = datasets.ImageFolder(root=data_dir)
```

Figura 3-30. Carga y preparación de la base de datos

La nueva arquitectura incorpora un mecanismo de *early stopping* (Figura 3-31), que funciona memorizando la mejor precisión de validación alcanzada por el modelo y comparándola con la del último *epoch*. Cuando la nueva precisión es peor que la más alta alcanzada por el modelo, una variable llamada “patience” o paciencia se encarga de parar el entrenamiento si la precisión no ha mejorado durante 7 *epochs*. Cuando se alcanza una nueva mejor precisión, se lleva a cabo el guardado de ese modelo entrenado en un archivo en formato .pth. De esta forma, se consigue que el entrenamiento se realice de forma más eficaz (al no detenerse hasta que la mejora de precisión se estabilice durante 7 *epochs*) y más eficiente, ya que no es necesario definir un número de *epochs* fijo antes de empezar a entrenar, sino solo un número límite.

```
# ===== LÓGICA DE EARLY STOPPING Y GUARDADO DEL MEJOR MODELO =====
if epoch_val_acc > best_acc:
    print(f" Nueva mejor precisión: {epoch_val_acc:.4f}. Guardando pesos...")
    best_acc = epoch_val_acc
    best_model_wts = copy.deepcopy(model.state_dict())
    epochs_no_improve = 0
else:
    epochs_no_improve += 1
    print(f" --> Sin mejora. Patience: {epochs_no_improve}/{patience}")

if epochs_no_improve >= patience:
    print(f"\nEARLY STOPPING en la época {epoch+1}. Volviendo al mejor modelo.")
    break

# --- FINALIZACIÓN ---
model.load_state_dict(best_model_wts) # Restaurar los mejores pesos encontrados
torch.save(model.state_dict(), 'best_cnn_modulation_model_STFT.pth')
print(f"\nEntrenamiento terminado. Mejor Accuracy de Validación: {best_acc:.4f}")
```

Figura 3-31. Mecanismo de *early stopping* y guardado de mejor modelo

3.4 Dataset en distribución de Choi-Williams

Se ha generado un segundo conjunto de datos compuesto por 64.000 imágenes empleando los mismos parámetros de generación de señales radar, pero utilizando Choi-Williams Distribution (CWD) como representación tiempo-frecuencia alternativa a STFT. La arquitectura modular de los *scripts* de MATLAB facilitó esta adaptación, requiriendo únicamente modificaciones puntuales en la subsección D (explicada en el apartado 3.2.3) de cada código, preservando intacta la lógica de síntesis de cada una de las modulaciones ya diseñadas.

La implementación de la CWD se llevó a cabo mediante la librería especializada *Time-Frequency Toolbox* (TFTB) para MATLAB, un conjunto no oficial de aproximadamente 100 *scripts* diseñado para el análisis de señales no estacionarias. Esta herramienta proporciona algoritmos eficientes para distribuciones cuadráticas de energía de la clase de Cohen, junto con funciones de generación de señales, procesamiento y visualización optimizadas para aplicaciones en procesamiento de señales radar. La librería ha sido descargada de [27] y ha sido instalada manualmente para poder emplearla en *Matlab*. De esta forma, no es necesario generar la señal en CWD de forma manual, lo que conllevaría iterar usando como punto de partida la Distribución de Wigner-Ville (WVD) y aplicando un filtro exponencial, elevando en gran medida el coste computacional.

La modificación de la subsección D se ha realizado conforme se ve en la Figura 3-32. Se ha fijado el parámetro de suavizado $\sigma=1$, para optimizar el equilibrio entre resolución tiempo-frecuencia y la supresión de términos cruzados. Su significado analítico es la configuración del núcleo exponencial de la función que filtra estas interferencias.

Se implementa la transformación tiempo-frecuencia mediante CWD utilizando la función “*tfrcw()*”, que es específica de la librería Time-Frequency Toolbox (TFTB). Esta función recibe como parámetros de entrada: la señal definida previamente (“*sig_slice*”), las posiciones temporales de análisis (“*1:num_muestras_win*”), el tamaño de la FFT (“*nfft*”) y el parámetro de suavizado (“*sigma*”), devolviendo la matriz de distribución tiempo-frecuencia (“*tfr*”). Posteriormente, se centra el espectro moviendo la componente de frecuencia cero al centro del espectrograma, generando la representación.

El post-procesamiento se realiza de la misma forma que en STFT, incluyendo la conversión a escala logarítmica, normalización y aplicación de umbral de suelo de ruido en -40 dB (Figura 3-9).

```
% --- D. GENERACIÓN DE CHOI-WILLIAMS ---
sigma = 1.0;
[tfr, ~, ~] = tfrcw(sig_slice, 1:num_muestras_win, nfft, sigma);

% Centrado de frecuencia
P = fftshift(abs(tfr), 1);

% Procesamiento de magnitud Logarítmica
img_data = 10*log10(P + eps);
img_data = img_data - max(img_data(:));
img_data(img_data < -40) = -40;
```

Figura 3-32. Código de generación de señal en CWD

3.5 Dataset en espectrograma STFT reasignado

Se ha generado también un tercer conjunto de datos compuesto por otras 64.000 imágenes con los mismos parámetros de generación, pero esta vez empleando representación STFT reasignado (véase el apartado 2.5.3). Para modificar la subsección D de los *scripts* de generación, se ha empleado la librería *Signal Processing Toolbox* de *Matlab*, que permite mediante “reassigned” el empleo de la función “*spectrogram()*” para generar STFT reasignado.

La modificación de la subsección D se ha realizado conforme se ve en la Figura 3-33. Se han establecido el tamaño de ventana y el solapamiento en los mismos valores que para generar la base de datos en STFT. La función “spectrogram()” se emplea de la misma manera, con la modificación de que se añade el parámetro “reassigned”, que activa el algoritmo de reasignación.

El post-procesamiento se realiza de la misma forma que en STFT, incluyendo la conversión a escala logarítmica, normalización y aplicación de umbral de suelo de ruido en -40 dB (Figura 3-9).

```
% --- D. GENERACIÓN DE ESPECTROGRAMA REASIGNADO ---
window_len = 32;
noverlap = floor(window_len * 0.90);
[~, ~, ~, P] = spectrogram(sig_slice, hamming(window_len), noverlap, nfft,
fs, 'centered', 'yaxis', 'reassigned');

% Procesamiento de magnitud
img_data = 10*log10(abs(P) + eps);
img_data = img_data - max(img_data(:));
img_data(img_data < -40) = -40;
```

Figura 3-33. Código de generación de señal STFT reasignado

3.6 Corrección de errores de generación y reentrenamiento (CNN-C)

Durante las pruebas de la red CNN-B se detectaron una serie de errores en la normalización de la imagen, así como en el mecanismo de adición de ruido (véase el apartado 4.2.4, en el que se explican los motivos de los errores en la generación de las imágenes de entrenamiento). Por ello se lleva a cabo una corrección de los *scripts* de generación de las bases de datos, llevándose a cabo modificaciones en las siguientes secciones:

- Método de normalización dinámica (subsección D, Figura 3-9).
- Mecanismo de adición de ruido (subsección C, Figura 3-8).
- Escalado de la imagen y selección del mapa de color (subsección E, Figura 3-10).

La normalización dinámica se sustituye por un intervalo de potencias fijo (Figura 3-34), calculado de forma empírica, generando y midiendo las señales. Se fuerza a todas las señales a estar dentro de ese intervalo, eliminando las diferencias de intensidad que aparecían al normalizar en función del valor máximo. Los intervalos establecidos para cada base de datos han sido:

- STFT: de -128 dB a -58 dB.
- CWD: de -55 dB a 30 dB.
- STFT reasignado: de -128 dB a -40 dB.

```
% --- D. GENERACIÓN DE ESPECTROGRAMA CENTRADO EN FRECUENCIA ---
window_len = 32;
noverlap = floor(window_len * 0.75);
[~, ~, ~, P] = spectrogram(sig_slice, hamming(window_len), noverlap, nfft,
fs, 'centered', 'yaxis');

% Procesamiento de magnitud
c_max = -58;
c_min = -128;
img_data = 10*log10(abs(P) + eps);
img_data(img_data > c_max) = c_max;
img_data(img_data < c_min) = c_min;
```

Figura 3-34. Modificación de la normalización de la imagen en STFT

El ruido se estaba generando a partir de la medición de un vector que había sido rellenado con ceros, por lo que la adición de ruido no estaba siendo adecuada. Para corregirlo (Figura 3-35), se mide la potencia de la señal mediante “bandpower ()”, se calcula el ruido necesario en función de la relación señal a ruido deseada y se suma a la señal.

```
% Ruido
power_sig = bandpower(sig);
power_noise = power_sig / (10^(snr/10));
noise = sqrt(power_noise) * (randn(num_muestras_win, 1)+...
1i*randn(num_muestras_win, 1))/sqrt(2);
sig_slice = sig_centered + noise;
```

Figura 3-35. Modificación de la adición de ruido

Por último, como se puede ver en la Figura 3-36, se elimina el mapa de color “jet”, que genera artefactos visuales y puede generar algunas distorsiones en la señal. Este mapa es sustituido mediante el empleo de “parula”, en el que los cambios de color están diseñados matemáticamente para que un incremento de 1 dB en la señal se perciba siempre como el mismo nivel de cambio visual. Además, se adelanta el redimensionamiento de la imagen para que se lleve a cabo antes de añadir el mapa de color, lo que ayuda a evitar la aparición de distorsiones en la señal.

```
% --- E. FORMATEO Y GUARDADO ---
img_norm = (img_data - c_min) / (c_max - c_min);
% Redimensionado
img_resized= imresize(img_norm, [518 518], 'bicubic');
img_final = ind2rgb(im2uint8(img_resized), parula(256));
```

Figura 3-36. Modificación del mapa de color y *resize*

Aplicando las modificaciones que se acaban de exponer, se genera una nueva base de datos para los espectrogramas elegidos (STFT, CWD y STFT reasignado) con 64.000 imágenes cada uno y se vuelve a entrenar la red neuronal (con la misma arquitectura), pasando a denominarla bajo el nombre de CNN-C.

4 RESULTADOS / VALIDACIÓN / PRUEBA

4.1 CNN-A

La red CNN-A se ha entrenado únicamente empleando la base de datos en STFT, que constaba de 64.000 imágenes, de las cuales el 80 % (51.200 imágenes) se han destinado a entrenamiento. En el apartado 0 se explica detalladamente el proceso de diseño del *dataset*.

El proceso de entrenamiento de la arquitectura CNN-A se ha realizado en 10 *epochs*, como se puede ver en la Figura 4-1, y se ha alcanzado una precisión final del 99,75 % en test, con un conjunto de datos independiente, del 99,84 % en validación y del 100 % en entrenamiento (Tabla 4-1).

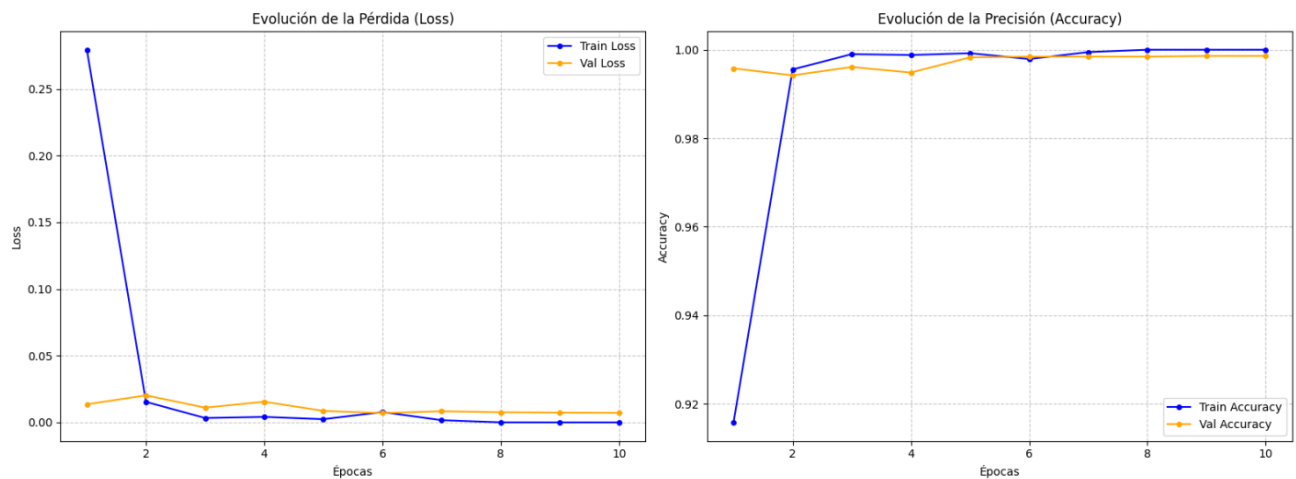


Figura 4-1. Curvas de aprendizaje CNN-A

Se puede observar como ya en el primer *epoch* la precisión de validación alcanza el 99,58 %, e incluso a partir del octavo *epoch* el *training loss* cae hasta llegar a valores nulos. A la vista de estos resultados, la red podría estar sufriendo *overfitting* o *data leakage*, ya que parece haber encontrado “atajos” en las imágenes para resolver el problema de identificación. La alta capacidad de la red para alcanzar el 100% de precisión de entrenamiento indica que la frontera de decisión es excesivamente simple para la complejidad real que debería tener el problema.

Training Accuracy	100 %
Validation Accuracy	99,84 %
Test Accuracy	99,75 %

Tabla 4-1. Precisión final de la CNN-A

Si se analiza la matriz de confusión normalizada (Figura 4-2), se observa que la identificación es, a la vista de los resultados, casi perfecta. El error más notable, aunque marginal, se produce entre las modulaciones Costas y SFH ya que el 1 % de las señales Costas son identificadas erróneamente como SFH. Esta confusión es coherente desde el punto de vista físico de la señal, debido a la similitud de estructuras de ambas señales, al tratarse ambas de modulaciones basadas en saltos de frecuencia.



Figura 4-2. Matriz de confusión CNN-A

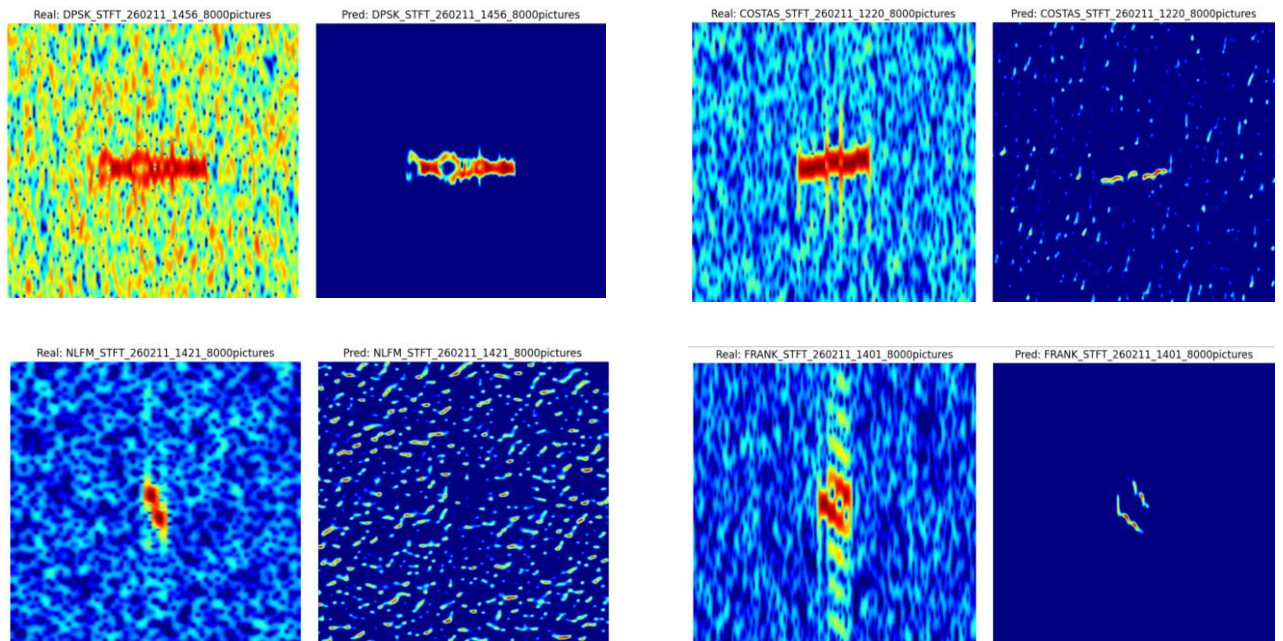


Figura 4-3. Análisis Grad-CAM de la CNN-A

El análisis visual de las activaciones mediante Grad-CAM muestra que la red se enfoca con gran precisión, en la mayoría de las ocasiones, en la morfología de la señal. Sin embargo, en entornos de SNR reducido la capacidad de la red de distinguir entre la señal y el ruido se ve limitada, como se puede ver en la señal NLFM de la Figura 4-3.

En resumen, la simplicidad de la arquitectura elegida para la red CNN-A y su alta dependencia de patrones cromáticos y de intensidad en los espectrogramas plantean dudas sobre su capacidad de generalización. Además, es probable que, en entornos con interferencias o elevado ruido, el rendimiento real se degrade significativamente, ya que la red ha aprendido a identificar "manchas" de color específicas en lugar de características robustas de la modulación.

4.2 CNN-B

En la Tabla 4-2 se muestra el resumen de características y resultados de las pruebas realizadas con la CNN-B tras su entrenamiento con bases de datos con espectrogramas en Transformada de Fourier (STFT), Distribución de Choi-Williams (CWD) y STFT reasignado.

	STFT	CWD	STFT reasignado
Test Accuracy	99,27 %	98,59 %	99,73 %
Val. Accuracy	99,50 %	98,70 %	99,81 %
Train. Accuracy	99,19 %	98,36 %	99,67 %
Peor modulación	P4 (confunde con Frank)	Costas (confunde con SFH)	P4 (confunde con Frank)
Bajo SNR	44,17 % a -8 dB	18,75 % a -8 dB	80,42 % a -8 dB

Tabla 4-2. Resumen de resultados de la CNN-B

4.2.1 STFT

El proceso de entrenamiento (Figura 4-4) de la arquitectura CNN-B con la base de datos generada en STFT ha requerido 38 *epochs*, ya que se ha empleado un procedimiento de *early stopping* (explicado

en el apartado 3.3). En la Figura 4-4 se pueden ver sus curvas de aprendizaje, guardándose automáticamente el modelo del *epoch* 31. Se ha alcanzado una precisión final del 99,27 % en test, con un conjunto de datos independiente, del 99,50 % en validación y del 99,19 % en entrenamiento (Tabla 4-3).

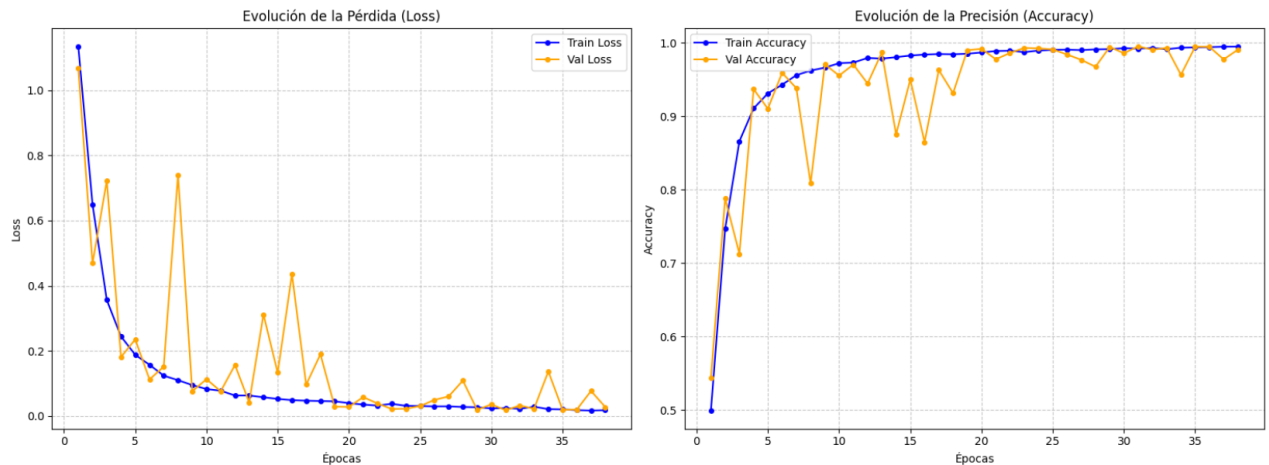


Figura 4-4. Curvas de aprendizaje CNN-B en STFT

La función de pérdida durante el entrenamiento (*Train Loss*) desciende suavemente hasta estabilizarse cerca de 0.0. La pérdida de validación (*Val Loss*) muestra algunas oscilaciones (picos de inestabilidad) en los primeras 20 *epochs*, pero eventualmente converge, lo que indica que el modelo no presenta aparentemente *overfitting*. En cuanto a la precisión (*Accuracy*), ambas curvas (entrenamiento y validación) superan el 90 % antes del *epoch* 5. La mejor precisión de validación se alcanza en el *epoch* 31, realizándose automáticamente el guardado de modelo en ese momento.

Training Accuracy	99,19 %
Validation Accuracy	99, 50%
Test Accuracy	99,27 %

Tabla 4-3. Accuracy final de la CNN-B en STFT

El análisis de la matriz de confusión muestra un acierto del 100 % en modulación Barker, Frank, NLFM y SFH. Los errores se producen en el resto, con porcentajes de acierto superiores al 98 %. El caso más notable es el de la modulación Costas, que se confunde en un 2 % de las ocasiones con SFH, al igual que en 4.1, debido a la similitud de ambos métodos al estar basados en saltos de frecuencia. Por otro lado, la otra fuente de confusión se produce en la predicción de imágenes P4, que confunde en un 1% de las ocasiones con LFM.

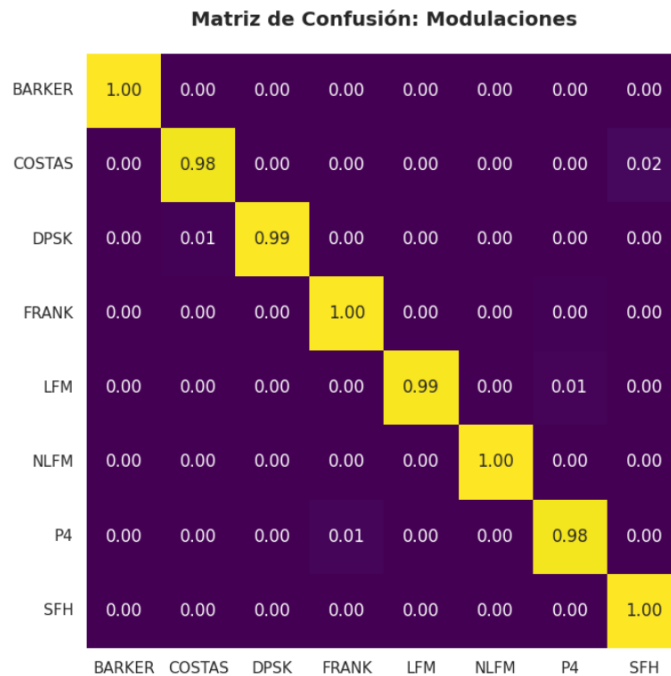


Figura 4-5. Matriz de confusión de la CNN-B en STFT

En la (Figura 4-6. Análisis Grad-CAM de la CNN-B en STFT) se pueden ver los resultados del análisis Grad-CAM de 8 predicciones en el test de la red. Las imágenes muestran que el modelo neuronal es capaz de extraer las características relevantes de las señales empleando espectrogramas STFT, incluso cuando la señal original presenta un ruido de fondo considerable (representado por las texturas verdosas/azuladas en las imágenes de entrada). El modelo es capaz de distinguir la señal en la representación para identificar el patrón de modulación central. El único punto débil aparente es el caso de la modulación P4, en la que distingue con dificultad las características más pequeñas de la señal (lo que lleva a la red a confundirla en ocasiones con LFM) y en Frank, en la que no centra su “visión” en la totalidad de la señal.

Sin embargo, se observan mejoras con respecto al modelo CNN-A, que pecaba de falta de profundidad y excesiva simpleza. A pesar de la ligera reducción de precisión, el modelo CNN-B presenta una arquitectura más compleja que debería ser capaz de generalizar con mejor resultado lo aprendido en un entorno real.

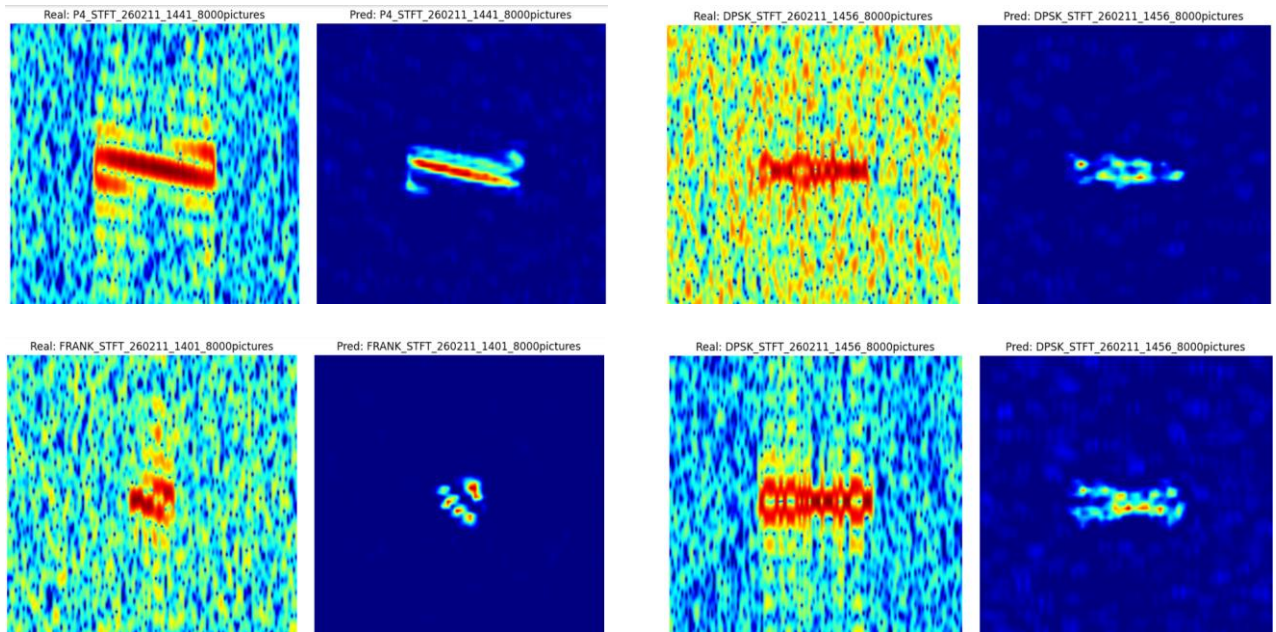


Figura 4-6. Análisis Grad-CAM de la CNN-B en STFT

4.2.2 CWD

El proceso de entrenamiento (Figura 4-7) de la arquitectura CNN-B con la base de datos generada mediante CWD ha requerido 42 *epochs*, empleándose igualmente un procedimiento de *early stopping* para evitar el sobreajuste. En la Figura 4-7 se muestran las curvas de aprendizaje, conservándose automáticamente el modelo correspondiente al *epoch* 36, en el que se alcanzó el mejor rendimiento de validación. La red logró una precisión final del 99,63 % en test, 99,72 % en validación y 99,55 % en entrenamiento (Tabla 4-4).

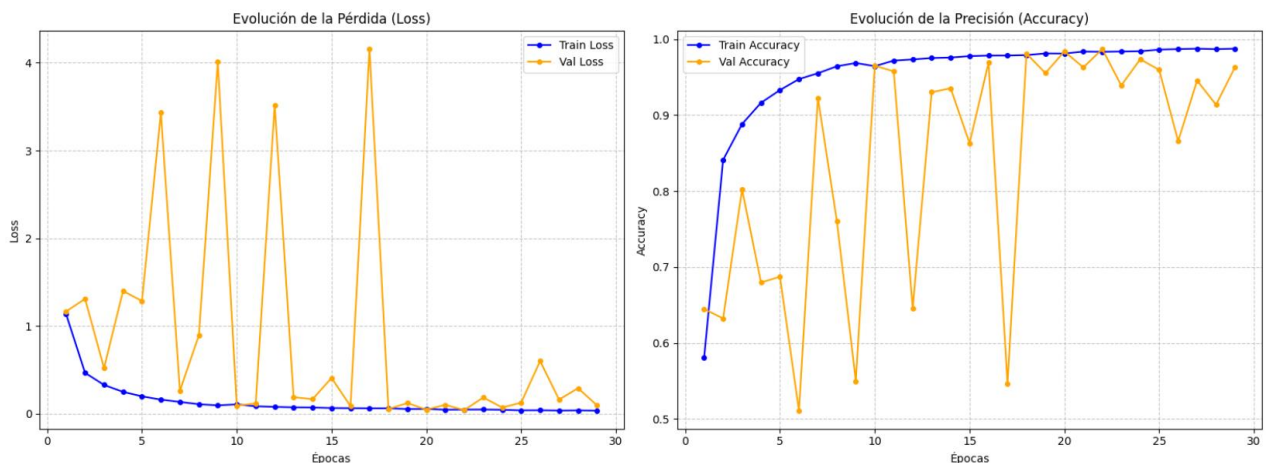


Figura 4-7. Curvas de aprendizaje CNN-B en CWD

Durante el entrenamiento, la función de pérdida (*Train Loss*) decrece de forma continua hasta estabilizarse en valores muy próximos a 0. La pérdida de validación (*Val Loss*) presenta importantes fluctuaciones durante los primeros 15 *epochs* (en concreto en el 6, 9, 12 y 17, en los que la precisión se reduce hasta rondar el 50 %), por lo que sería interesante reducir la tasa de aprendizaje para el entrenamiento con la base de datos en CWD. A partir del *epoch* 18, converge de forma más estable, lo que indica que el modelo consigue finalmente una buena capacidad de generalización sin signos evidentes de *overfitting*. En cuanto a la precisión (*accuracy*), tanto la de entrenamiento como la de validación superan el 90 % antes del *epoch* 4, aunque en validación se vuelven a ver las mencionadas

fluctuaciones. La mejor precisión de validación se obtiene en el *epoch* 36, en el cual se lleva a cabo el guardado automático del modelo.

Training Accuracy	98,36 %
Validation Accuracy	98,70 %
Test Accuracy	98,59 %

Tabla 4-4. Precisión final de la CNN-B en CWD

El análisis de la matriz de confusión (Figura 4-8) muestra una tasa de acierto del 100 % de las modulaciones Barker, DPSK, Frank y SFH. Mientras que las mayores dificultades se producen con modulaciones Costas, en las que la precisión baja al 94 %. Esta modulación es confundida principalmente con SFH, lo cual se producía también en el entrenamiento con STFT debido a la similitud de estas modulaciones; pero además se producen errores con DPSK. La red tiene un 98 % de precisión en las señales del tipo P4, que confunde en un 2 % de las ocasiones con LFM, al ser ambos patrones espectrales muy similares.

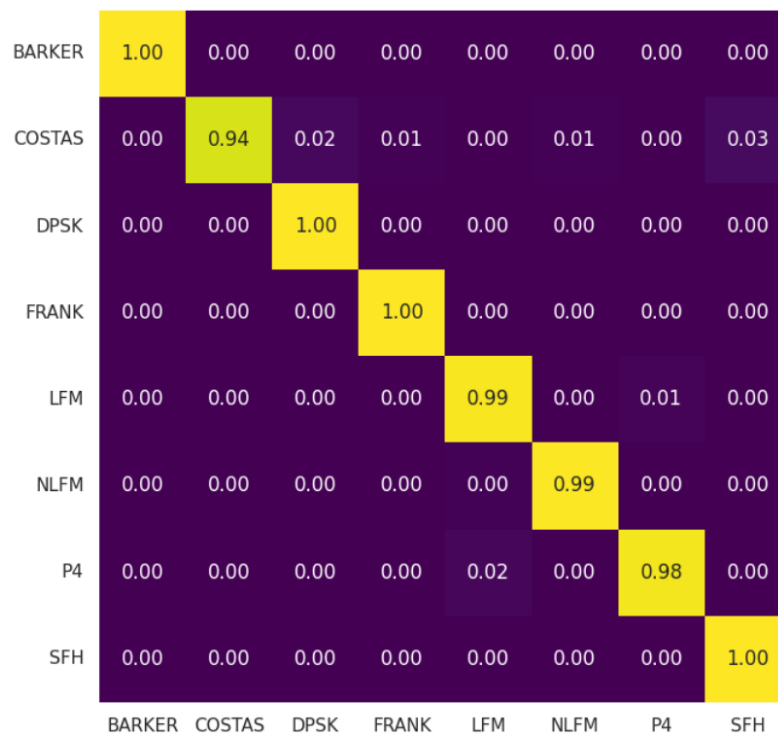


Figura 4-8. Matriz de confusión de la CNN-B en CWD

En la Figura 4-9 se pueden observar los resultados del análisis Grad-CAM de 4 predicciones en el conjunto de test. Las visualizaciones evidencian que el modelo identifica de forma precisa las regiones del espectrograma CWD asociadas a la señal radar en la gran mayoría de las ocasiones. Se aprecia además una mayor concentración de activación en las zonas relevantes de la señal, incluso bajo presencia de ruido o artefactos menores en el entorno espectral. No obstante, en caso de la modulación SFH, el modelo neuronal tiende a fijarse en el ruido, que no consigue diferenciar en su totalidad de la señal radar, a pesar de conseguir hacer una predicción correcta.

El modelo CNN-B entrenado en CWD presenta un ligero empeoramiento con respecto al modelo entrenado en STFT, que muestra curvas de aprendizaje más estables y una precisión ligeramente mejor.

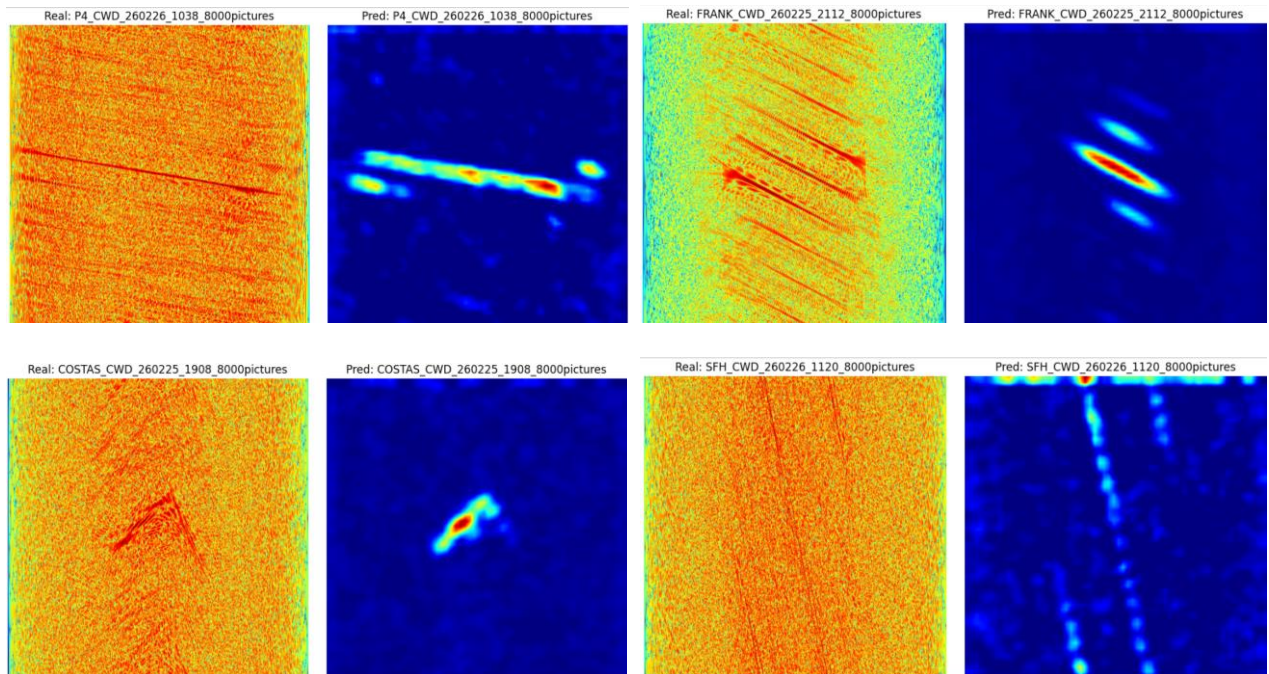


Figura 4-9. Análisis Grad-CAM de la CNN-B en CWD

4.2.3 STFT reasignado

El proceso de entrenamiento de la arquitectura CNN-B con la base de datos generada en STFT reasignado se ha extendido hasta los 38 *epochs*. Las curvas de aprendizaje se muestran en la Figura 4-10, habiéndose realizado el guardado automático del modelo en el *epoch* 31 (precisión de validación 99,81%). Se alcanza una precisión final del 99,73% en test, y del 99,67 % en entrenamiento, como se resume en la Tabla 4-5.

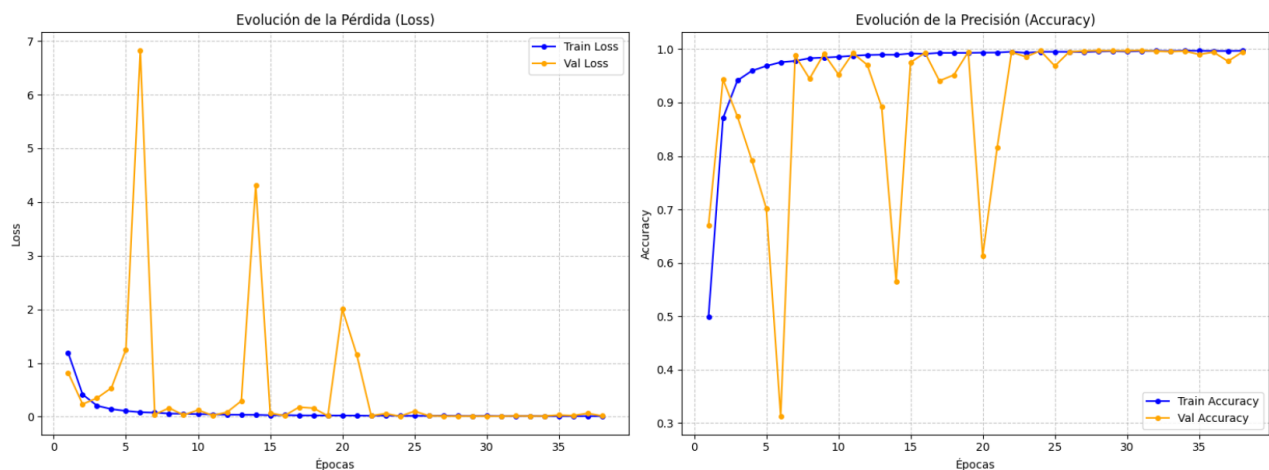


Figura 4-10. Curvas de aprendizaje CNN-B en STFT reasignado

La función de pérdida de entrenamiento (*Train Loss*) desciende de manera progresiva y estable hasta valores inferiores a 0,01. La pérdida de validación (*Val Loss*) sufre oscilaciones pronunciadas en etapas tempranas (*epochs* 4-7, 14-20) con picos aislados que alcanzan valores elevados (hasta 6,8), pero converge definitivamente hacia 0,007-0,01 en las últimas fases y sin overfitting aparente. Las curvas de precisión (*accuracy*) muestran un ascenso rápido inicial (superior al 95% antes del *epoch* 10), con fluctuaciones menores en validación que se estabilizan superando el 99,5% desde el *epoch* 28. La mejor validación se logra en el *epoch* 31, llevándose a cabo el guardado automático de ese modelo.

Training Accuracy	99,67 %
Validation Accuracy	99,81 %
Test Accuracy	99,73 %

Tabla 4-5. Precisión final de la CNN-B en STFT reasignado

El análisis de la matriz de confusión muestra un rendimiento casi perfecto, con 100% de acierto en Barker, Costas, DPSK, Frank, LFM, NLFM y SFH. El único error registrado es de un 1% en P4, confundida marginalmente con Frank, debido probablemente a las similitudes en la estructura polifásica de ambas modulaciones en la representación elegida para el entrenamiento.

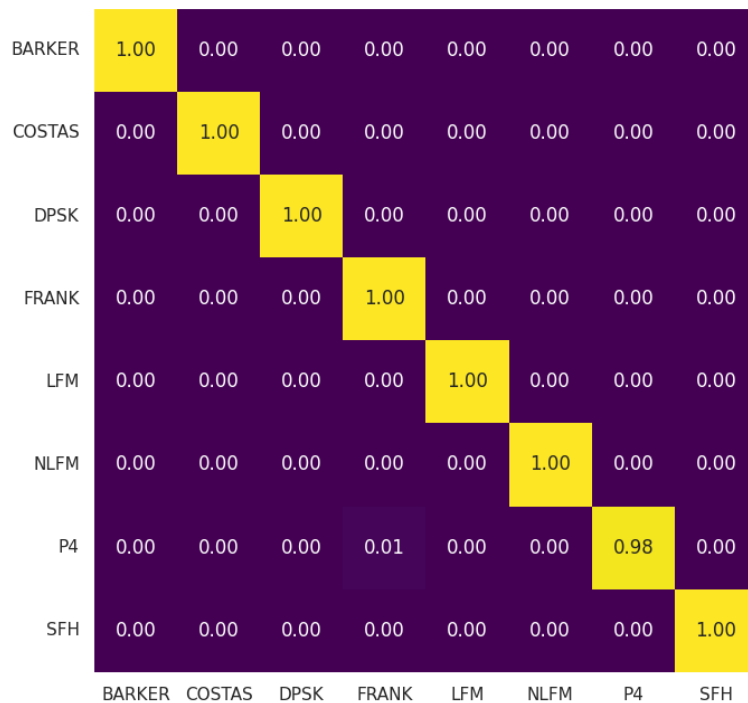


Figura 4-11. Matriz de confusión de la CNN-B en STFT reasignado

En la Figura 4-12 se presentan resumidos los resultados del análisis Grad-CAM sobre las predicciones del test, incluyendo ejemplos de LFM, Frank, P4 y DPSK. Las activaciones destacan la capacidad del modelo para focalizar su “visión” en las características de la señal radar, ignorando con gran precisión el ruido de fondo y capturando patrones finos como las transiciones polifásicas en P4 o las lineales en LFM. Se observa una concentración precisa en toda la extensión de la señal para Frank (mejorando las limitaciones vistas en STFT/CWD), aunque en P4 el foco podría intensificarse en bordes periféricos para eliminar el 1% residual de confusión.

A pesar de mantener dificultades iniciales en el entrenamiento de la red, con grandes fluctuaciones de la función de pérdida en validación, el modelo ha alcanzado la mayor precisión en su entrenamiento con la base de datos en STFT reasignado. Además, la red ha demostrado una alta capacidad de discriminación del ruido de fondo.

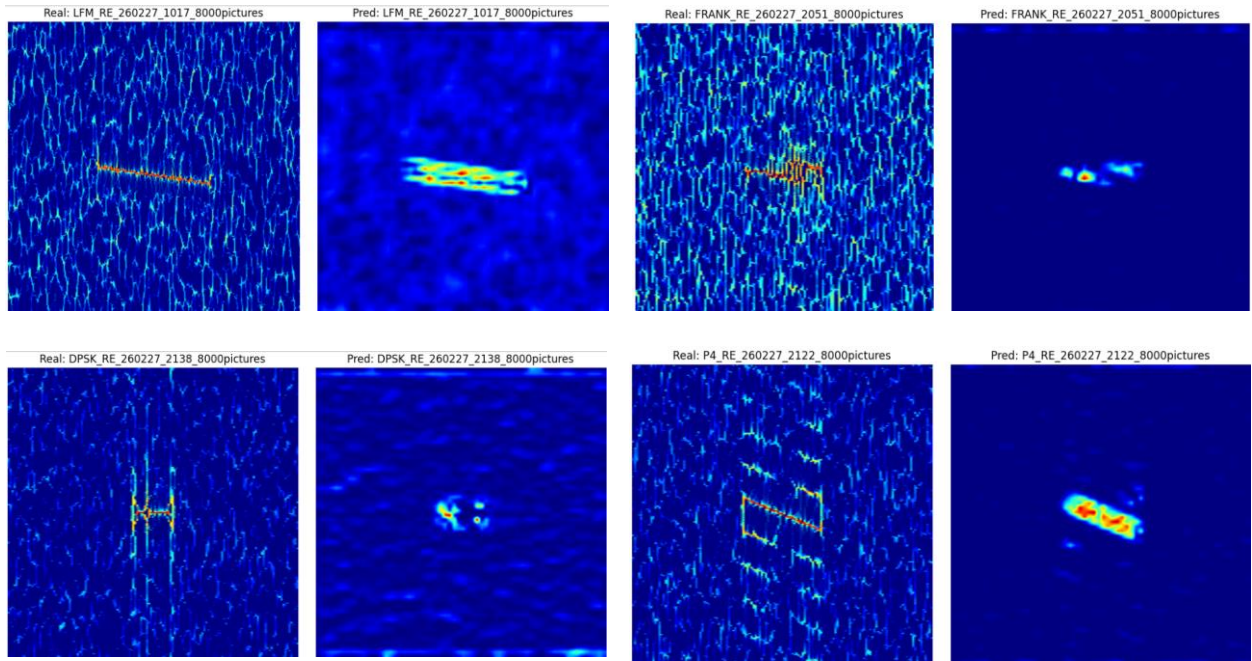


Figura 4-12. Análisis Grad-CAM de la CNN-B en STFT reasignado

4.2.4 Robustez frente a reducción de SNR del modelo CNN-B

Para comprobar la capacidad de los modelos CNN-B de generalizar en entornos de bajo SNR se han generado 240 imágenes (30 por modulación) para cada SNR del intervalo seleccionado (desde -20 dB hasta 8 dB en saltos de 2 dB). De esta forma, se ha medido la precisión del modelo para cada SNR dando los resultados que se muestran en la Figura 4-13.

Estas pruebas se han realizado con SNR que bajan de -10 dB, para evaluar la degradación de la precisión que sufre la red. En [28] usan una CNN para realizar el *denoising* de señales LFM, consiguiendo tasas de error aceptables con SNR por encima de -10 dB.

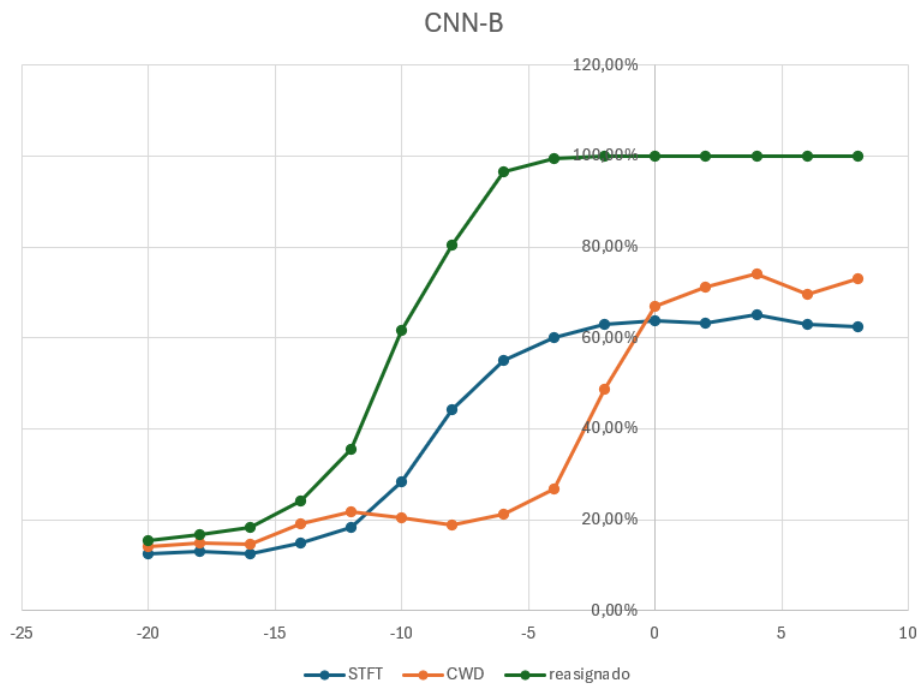


Figura 4-13. Resumen de resultados con bajo SNR

Los resultados obtenidos por los diferentes modelos neuronales en los intervalos para los que han sido entrenados se muestran resumidos en la Tabla 4-6.

El modelo neuronal entrenado con la base de datos en STFT ha conseguido una precisión entre el 55 % y el 62,5 % en el intervalo de SNR en el que fue entrenado (-6 dB a 8 dB), dando una media de 61,92 %. Siendo unos resultados mucho más bajos que los obtenidos durante la fase de test inicial (4.2.1), que mostraban una precisión del 99,27 %. Por otro lado, a partir de -6 dB la precisión se hunde, pasando del 44,1 % en -8 dB al 28,3 % en -10 dB.

En el caso de la red entrenada con la base de datos generada en CWD, la precisión obtenida varía entre el 21,25 % y el 72,92 % en el intervalo de SNR de -6 dB a 8 dB, quedando la media en 56,45 %. Los resultados no solo son notablemente más bajos que los obtenidos durante la fase de test inicial (4.2.2), sino que la precisión se desploma desde los -4 dB, manteniéndose por debajo del 30 %. La precisión máxima (72,92 %) es mayor que la conseguida en STFT, sin embargo, el modelo se muestra mucho más inestable en esta configuración, con variaciones de precisión erráticas.

	STFT	CWD	STFT reasignado
Intervalo de entrenamiento (-6 dB a 8 dB)	55 % - 62,5 %	21,25 % - 72,92 %	96,67 % - 100 %
Media intervalo de entrenamiento	61,92 %	56,45 %	99,53 %

Tabla 4-6. Resumen resultados robustez a bajo SNR

El único modelo que ha mantenido la precisión mostrada en la fase inicial de test (véase el apartado 4.2.3) ha sido la CNN-B entrenado con STFT reasignado. Esta red ha conseguido una precisión en el intervalo de entrenamiento que varía entre el 96,67 % y el 100 %, dando una media de 99,53 %. Este resultado es ligeramente inferior a la precisión de test inicial del 99,73 %. Conforme se reduce la relación señal a ruido, la precisión se reduce progresivamente: en -8 dB la red es capaz de acertar en el 80,42 % de las ocasiones; a partir de -10 dB la precisión cae por debajo del 60 %.

La red en STFT ha obtenido un rendimiento muy por debajo del obtenido en el entrenamiento de la red, demostrando que no ha sido capaz de generalizar a partir de las imágenes usadas en su entrenamiento. La red CNN-B entrenada con imágenes en CWD tiene el rendimiento más deficiente de los 3 modelos, registrando apenas un 20 % de precisión a -8 dB, gran inestabilidad y un rendimiento bajo en el intervalo en el que fue entrenada.

Para averiguar lo ocurrido, se extraen las matrices de confusión a 0 dB (Figura 4-14), en la se puede observar que la red es capaz de acertar 5 de las 8 modulaciones con precisiones elevadas, mientras que en el resto (Barker, NLFM y SFH) falla entre el 87 % y el 100 % de las predicciones. En el caso de CWD, la falta de precisión se concentra en NLFM, Frank y Costas, aunque aún es capaz de acertar en más de un 80 % de los casos Barker, DPSK, LFM y SFH.

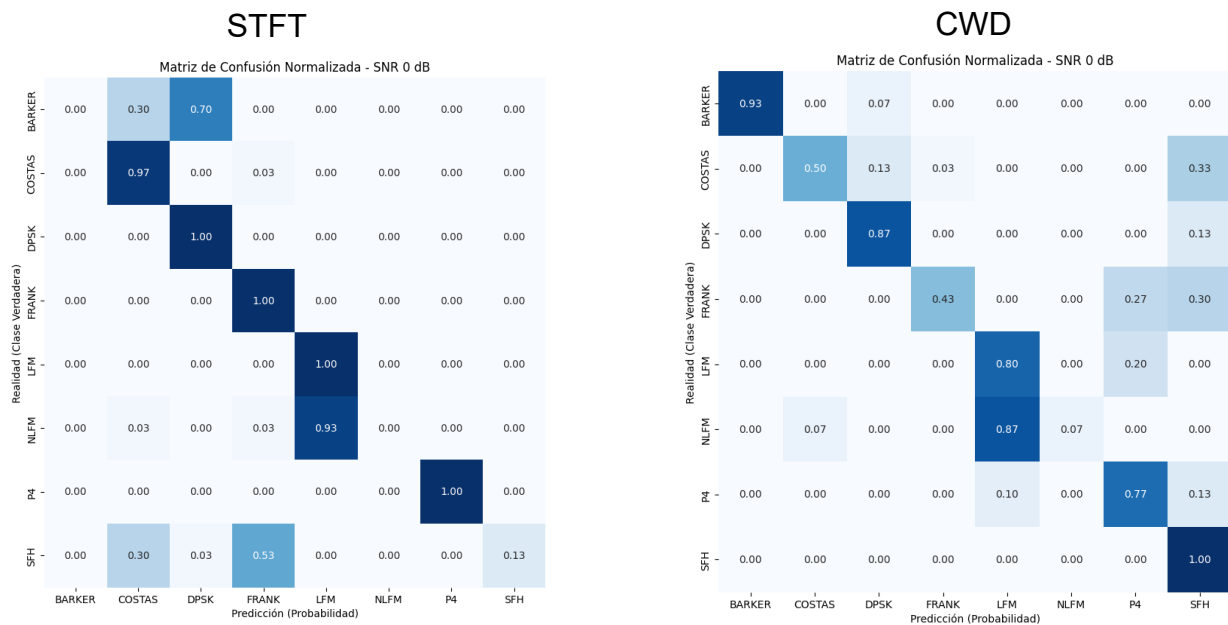


Figura 4-14. Matriz de confusión de las pruebas de bajo SNR en STFT

Además, se lleva a cabo un análisis Grad-CAM (Figura 4-15), el cual revela que, en las modulaciones en las que las redes fallan con asiduidad, la visión artificial se fija en algunos de los bordes de la señal, pero también en el ruido; es decir, la red aún es capaz de detectar las características morfológicas (por los cambios de intensidad entre píxeles), pero duda entre el ruido y la propia señal radar. Esto lleva a pensar que el problema no está derivado de un *data leakage* por falta de estanqueidad, o un error distinto en la división de las bases de datos, sino que puede estar en la propia generación de las imágenes, existiendo una diferencia entre las imágenes generadas para entrenamiento y las de las pruebas de reducción de la SNR.

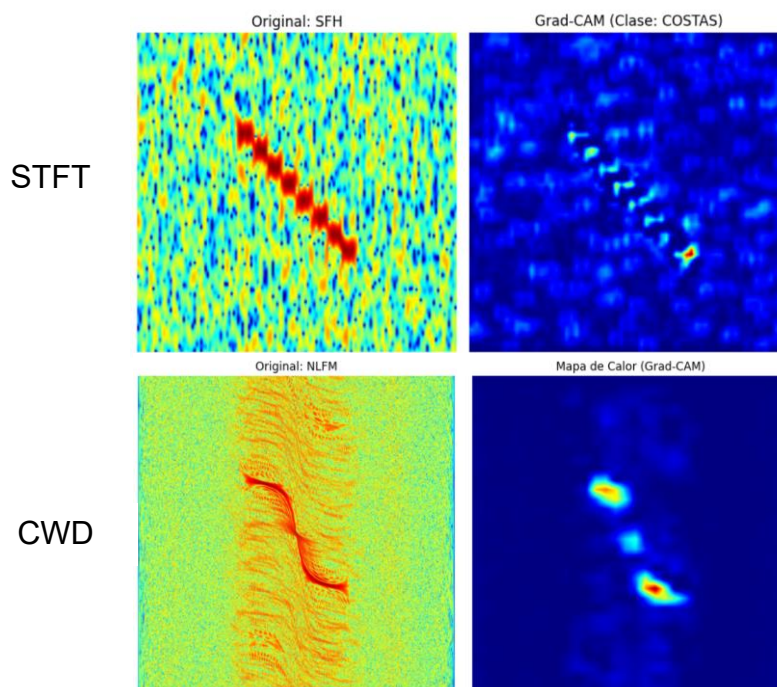


Figura 4-15. Análisis Grad-CAM de las imágenes de la prueba de bajo SNR

Para comprobar que el error en la generación de la base de datos está provocando la disminución del rendimiento, se lleva a cabo un análisis de la distribución de intensidades (Figura 4-16) de las imágenes generadas para la base de datos original y las generadas posteriormente para realizar las pruebas. Los resultados muestran que la red neuronal está sufriendo de *data drift* o *covariance shift*, es decir, las imágenes de entrenamiento son más “brillantes” o con colores más vivos que las de prueba. Esto puede llegar a afectar a la activación de algunas funciones, como *ReLU*, haciendo que alguna de las funciones de activación *ReLU* de cero como salida. De esta forma, la red ha aprendido una serie de reglas o patrones que en las imágenes de la prueba no le sirven.

Este fallo lo provoca la línea de código (Figura 3-9) encargada de la normalización dinámica, en la que se establecía el techo de energía de 0 dB en función del valor máximo, y después se limitaba el suelo de ruido en un valor fijo. Para corregirlo se debe establecer un intervalo de normalización fijo (véase el apartado 3.6), lo que permitirá que todas las imágenes se comparen en igualdad de condiciones.

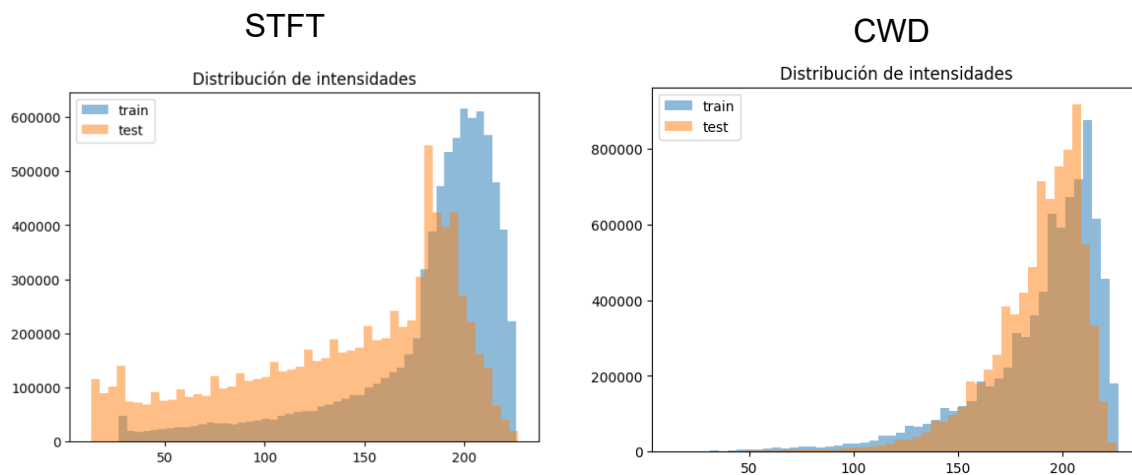


Figura 4-16. Distribución de intensidades de las imágenes STFT y CWD

En la revisión del código de generación se detectan otros errores (véase el apartado 3.6), como la adición del mapa de colores antes del escalado de la imagen mediante “jet”. Este mapa de color puede generar artefactos y deformaciones, por lo que debe sustituirse y emplearse después del escalado. El segundo error detectado deriva de la adición de ruido gaussiano (función “awgn” de *Matlab*), que se realizaba de forma incorrecta, por lo que también se debe rediseñar.

Sin embargo, el modelo entrenado con el espectrograma reasignado es el único que no solo ha demostrado una robustez adecuada, sino que es capaz de mantener una precisión adecuada en el intervalo de SNR en el que ha sido entrenado. Como se aprecia en las gráficas de precisión (Figura 4-13), este modelo mantiene una curva de rendimiento prácticamente plana en el rango de entrenamiento, conservando una precisión del 80,42% incluso a -8 dB.

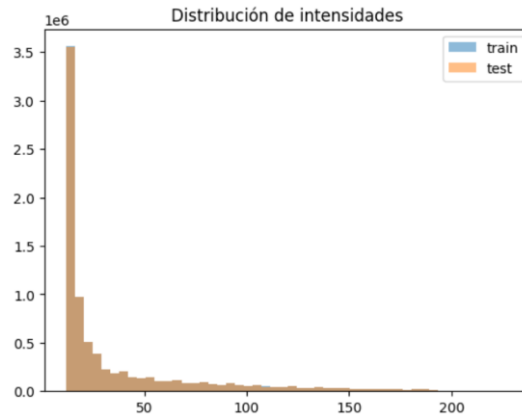


Figura 4-17. Distribución de intensidades de las imágenes STFT reasignado

El éxito de este modelo se debe a la propia definición del método STFT reasignado, descrita en el apartado 2.5.3, empleando una técnica de reasignación de energía en lo que denomina centro de gravedad de la señal. Este proceso genera representaciones visuales de una elevada precisión, convirtiendo las modulaciones en estructuras lineales nítidas; mientras que la STFT normal y la CWD “emborronan” la energía de la señal, creando gradientes suaves. Este es el motivo por el que el histograma de intensidad tiene forma de campana, al existir muchos pixeles con valores intermedios.

La STFT Reasignada, en cambio, concentra toda la energía en puntos exactos de tiempo y frecuencia, lo que se puede observar en la distribución de intensidades (Figura 4-17), en la que no hay apenas diferencias entre las imágenes de entrenamiento y prueba. De esta forma se obtienen dos ventajas: el fondo de la imagen (ruido) tiene muy poca energía y la propia señal tiene la energía concentrada en una línea de ancho muy reducido. Es decir, este espectrograma aplanar el ruido y afila la señal, lo que la hace prácticamente inmune al error producido en la generación de la base de datos sintética.

4.3 CNN-C en STFT y CWD

En la Tabla 4-7 se muestra el resumen de características y resultados de las pruebas realizadas con la CNN-C tras su entrenamiento usando las bases de datos con espectrogramas en Transformada de Fourier (STFT), Distribución de Choi-Williams (CWD) y STFT reasignado.

	STFT	CWD
Test Accuracy	94,00 %	92,33 %
Val. Accuracy	93,09 %	92,89 %
Train. Accuracy	94,84 %	92,25 %
Peor modulación	LFM (confunde con P4)	Costas Y Barker (confunde con DPSK)
Comportamiento con bajo SNR	Regular: 51,67 % a -6 dB	Malo: 50,42 % a 0 dB

Tabla 4-7. Resumen de resultados CNN-C

4.3.1 STFT

El proceso de entrenamiento (Figura 4-18) de la arquitectura CNN-C con la nueva base de datos generada en STFT ha requerido 74 *epochs*, aplicándose un procedimiento de *early stopping* que detiene el entrenamiento cuando la precisión de validación deja de mejorar durante 10 iteraciones. El mejor

modelo guardado ha sido el correspondiente al *epoch* 64, en el que la red alcanza unas precisiones (Tabla 4-8) del 94,84 % en entrenamiento, 93,09 % en validación y 94,00 % en el test con la parte del *dataset* reservada para ello.

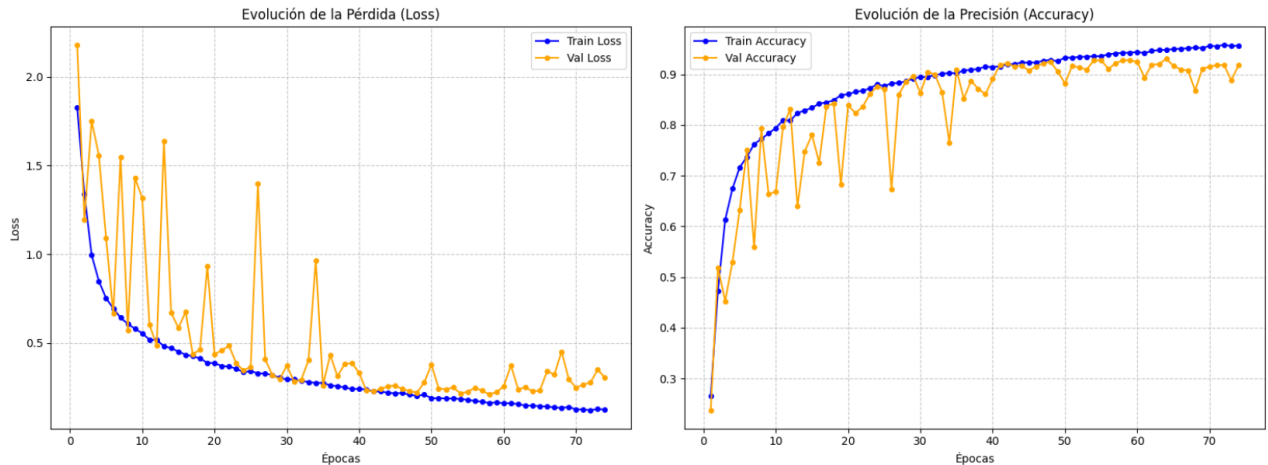


Figura 4-18. Curvas de aprendizaje CNN-C en STFT

La función de pérdida durante el entrenamiento (*Train Loss*) muestra un descenso progresivo, pasando de valores superiores a 1.8 en las primeras iteraciones a valores cercanos a 0.15 en las últimas fases del entrenamiento. La pérdida de validación (*Val Loss*) presenta un comportamiento más irregular, con oscilaciones significativas durante buena parte del entrenamiento, especialmente en los primeros 30 *epochs*. A partir de las últimas fases del entrenamiento, la pérdida de validación tiende a estabilizarse alrededor de valores cercanos a 0.22–0.23, coincidiendo con los mejores resultados de precisión.

En cuanto a la precisión (*Accuracy*), tanto la curva de entrenamiento como la de validación muestran un incremento progresivo, si bien, en las primeras iteraciones, el modelo parte de precisiones relativamente bajas (alrededor del 25–50 %), pero alcanzando finalmente el 75 % antes del *epoch* 10. A partir de ese punto, la mejora continúa de forma gradual hasta alcanzar su mejor precisión de validación en el *epoch* 64 (guardado automático de mejor modelo).

Training Accuracy	94,84 %
Validation Accuracy	93,09 %
Test Accuracy	94,00 %

Tabla 4-8. Accuracy final de la CNN-C en STFT

El análisis de la matriz de confusión (Figura 4-19) muestra que, en general, la red consigue altos porcentajes de acierto en la mayoría de las modulaciones, aunque se observan ciertas confusiones entre aquellas técnicas que presentan características espectrales similares en la representación STFT. En este caso, el modelo tiene dificultades en Costas (91 % de acierto), que confunde con SFH; y en P4 (92 % de acierto), que no distingue en un 7 % de las ocasiones con Frank y LFM. Otro de los fallos recurrentes se produce en LFM (91 % de acierto), confundiéndola en el 7 % de las ocasiones con P4.

Matriz de Confusión: STFT

BARKER	0.96	0.01	0.02	0.01	0.00	0.00	0.00	0.00
COSTAS	0.02	0.91	0.01	0.01	0.00	0.01	0.00	0.04
DPSK	0.04	0.01	0.94	0.01	0.00	0.00	0.00	0.00
FRANK	0.00	0.00	0.01	0.98	0.00	0.00	0.01	0.00
LFM	0.00	0.00	0.00	0.01	0.91	0.01	0.07	0.00
NLFM	0.00	0.00	0.00	0.01	0.01	0.96	0.01	0.00
P4	0.00	0.00	0.00	0.04	0.03	0.00	0.92	0.00
SFH	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.95
	BARKER	COSTAS	DPSK	FRANK	LFM	NLFM	P4	SFH

Figura 4-19. Matriz de confusión CNN-C en STFT

En la Figura 4-20 se muestran los resultados del análisis Grad-CAM aplicado a diversas predicciones del conjunto de test. Los mapas de calor indican que la red es capaz de centrar su atención en las regiones del espectrograma que contienen la estructura principal de la señal, ignorando en gran medida el ruido de fondo presente en la representación STFT. En algunos casos, no obstante, la activación se distribuye de forma más difusa entre la señal y el ruido, lo que podría explicar algunas de las confusiones en modulaciones con estructuras más complejas o con menor contraste respecto al ruido.

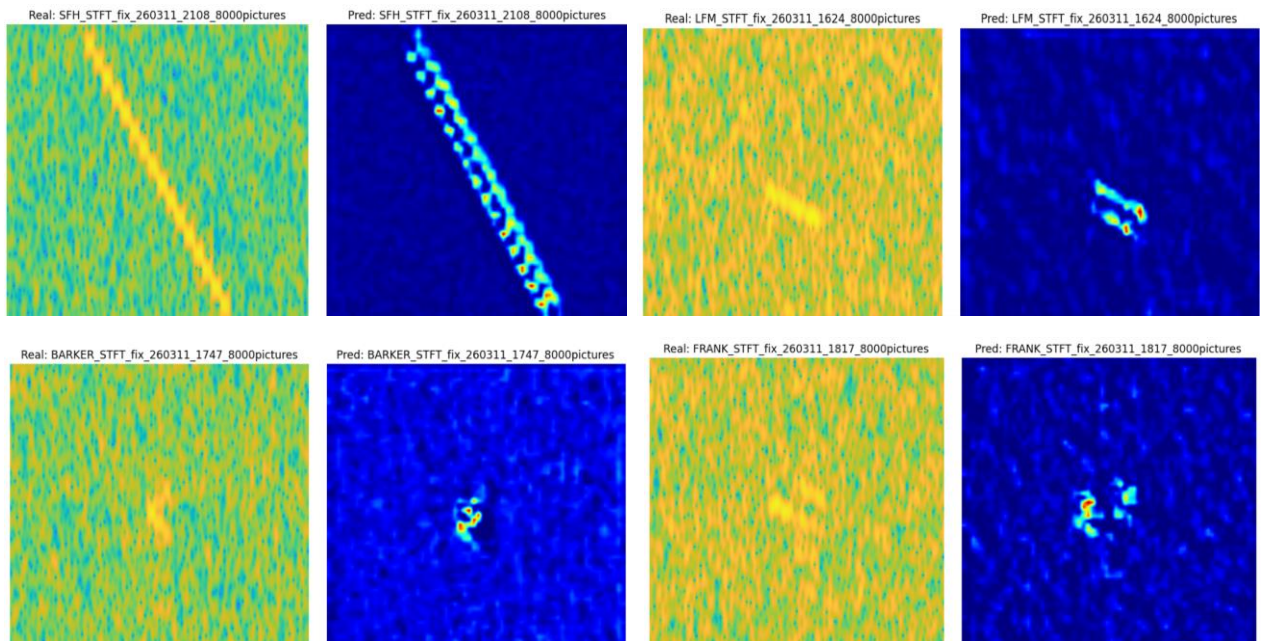


Figura 4-20. Análisis Grad-CAM de CNN-C en STFT

4.3.2 CWD

El entrenamiento del modelo CNN-C en CWD (Figura 4-21) ha requerido un total de 33 *epochs*, aplicándose un procedimiento de *early stopping* de 10 *epochs* de “patience” o paciencia (explicado en

el apartado 3.3). El mejor modelo guardado ha sido el correspondiente al *epoch* 23, en el que la red alcanza unas precisiones (Tabla 4 8) del 92,33 % en entrenamiento, 92,89 % en validación y 92,25 % en el test con la parte del *dataset* reservada para ello.

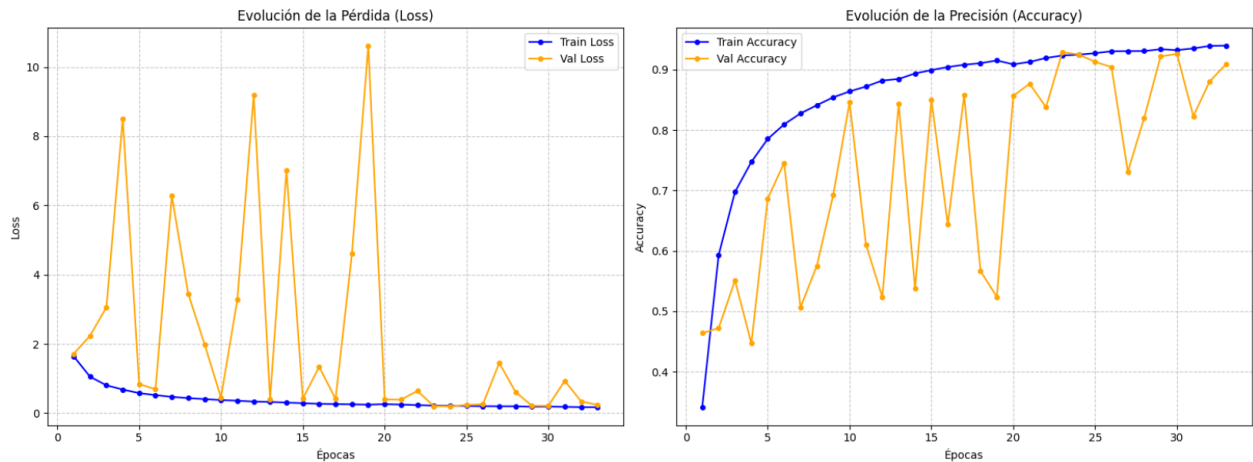


Figura 4-21. Curvas de aprendizaje CNN-C en CWD

La función de pérdida durante el entrenamiento (*Train Loss*) muestra un descenso progresivo, pasando de valores iniciales cercanos a 1.64 en las primeras iteraciones a valores próximos a 0.17–0.18 en las últimas fases del entrenamiento. La pérdida de validación (*Val Loss*) presenta un comportamiento considerablemente más irregular, con oscilaciones significativas a lo largo de buena parte del entrenamiento. En varios *epochs* se observan picos elevados de pérdida de validación, lo que indica que el modelo puede experimentar episodios puntuales de sobreajuste. No obstante, en torno a los *epochs* 20–25 la pérdida de validación alcanza sus valores mínimos (alrededor de 0.20), coincidiendo con los mejores resultados de precisión del modelo.

En cuanto a la precisión (*Accuracy*), tanto la curva de entrenamiento como la de validación muestran un incremento progresivo a lo largo de las iteraciones. El modelo parte de valores relativamente moderados en el primer *epoch* (alrededor del 34 % en entrenamiento y 46 % en validación), pero experimenta una mejora rápida durante las primeras iteraciones. Antes del *epoch* 10, la red ya supera el 85 % de precisión. La red alcanza su mejor precisión de validación en el *epoch* 23, en el que se lleva a cabo el guardado automático del mejor modelo.

Training Accuracy	92,33 %
Validation Accuracy	92,89 %
Test Accuracy	92,25 %

Tabla 4-9. Accuracy final CNN-C en CWD

La matriz de confusión (Figura 4-22) muestra que la red tiene una distribución de la precisión homogénea. No obstante, se observan ciertas confusiones entre modulaciones con patrones tiempo-frecuencia similares, siendo la modulación con menos aciertos (86 % de precisión) las señales Costas, que confunde con DPSK. Otro caso notable es el de la modulación Barker, que es confundida una de cada diez veces con DPSK. Estas confusiones se concentran principalmente entre modulaciones cuya distribución energética en el plano tiempo-frecuencia comparte trayectorias o estructuras similares, siendo más acuciantes que las vistas en la red entrenada con espectrograma STFT (véase el apartado 4.3.1).

BARKER	0.90	0.00	0.10	0.00	0.00	0.00	0.00	0.00
COSTAS	0.02	0.86	0.07	0.00	0.00	0.01	0.00	0.04
DPSK	0.03	0.02	0.94	0.00	0.00	0.00	0.00	0.00
FRANK	0.01	0.00	0.02	0.96	0.00	0.00	0.00	0.00
LFM	0.00	0.00	0.00	0.01	0.91	0.02	0.04	0.01
NLFM	0.01	0.02	0.01	0.00	0.00	0.95	0.00	0.01
P4	0.00	0.00	0.00	0.04	0.05	0.00	0.90	0.00
SFH	0.01	0.01	0.02	0.00	0.00	0.00	0.00	0.96
	BARKER	COSTAS	DPSK	FRANK	LFM	NLFM	P4	SFH

Figura 4-22. Matriz de confusión de la CNN-C en CWD

Los resultados del análisis Grad-CAM se muestran en la Figura 4-23 y corresponden a las imágenes de test. Los mapas de calor indican que la red es capaz de centrar su atención en las regiones del plano tiempo-frecuencia donde se concentra la energía de la señal, identificando las estructuras características de cada modulación. En la mayoría de los casos, la activación se concentra claramente sobre las trazas principales de la señal en la representación CWD, mientras que el ruido de fondo recibe una activación considerablemente menor. No obstante, en algunas predicciones la activación aparece más difusa o distribuida, lo que podría explicar ciertos errores de clasificación cuando las estructuras de la señal presentan menor contraste o mayor complejidad.

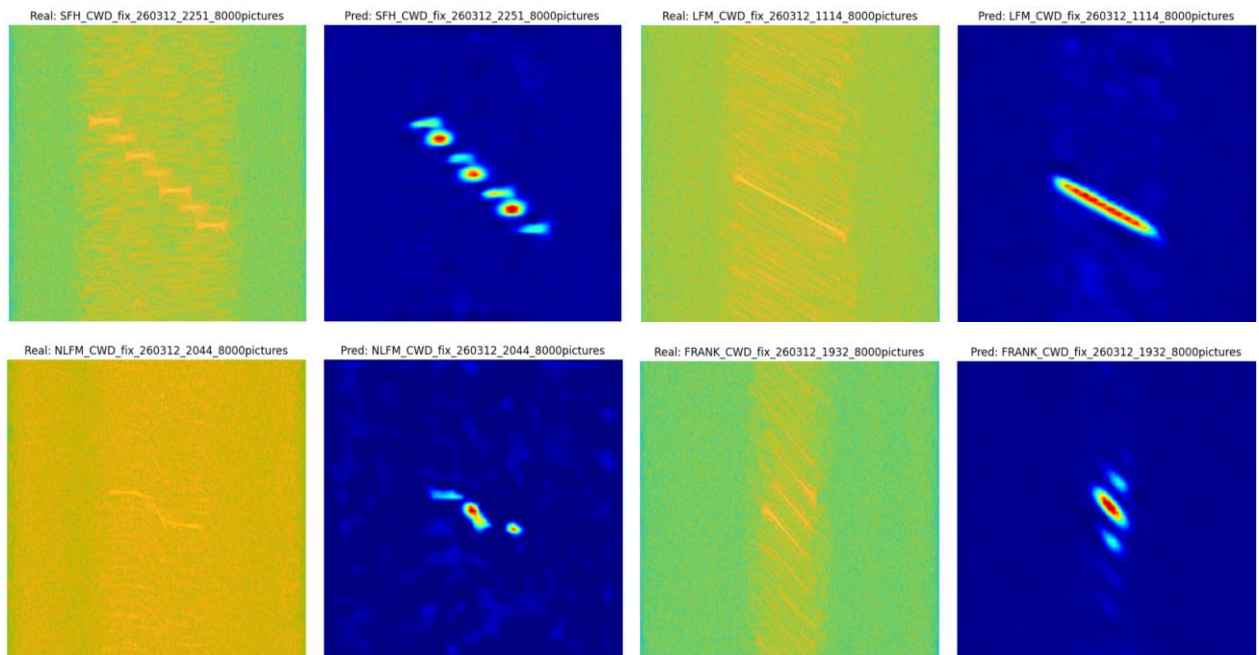


Figura 4-23. Análisis Grad-CAM de la CNN-C en CWD

4.4 Robustez frente a reducción de SNR de las CNN definitivas

Para comprobar la capacidad de los modelos CNN definitivos (CNN-C en STFT, CNN-C en CWD y CNN-B en STFT) de generalizar en entornos de bajo SNR se han generado nuevamente 240 imágenes (30 por modulación) para cada SNR del intervalo seleccionado (desde -20 dB hasta 8 dB en saltos de 2 dB). De esta forma, se ha medido la precisión del modelo para cada SNR dando los resultados que se muestran en la Figura 4-24.

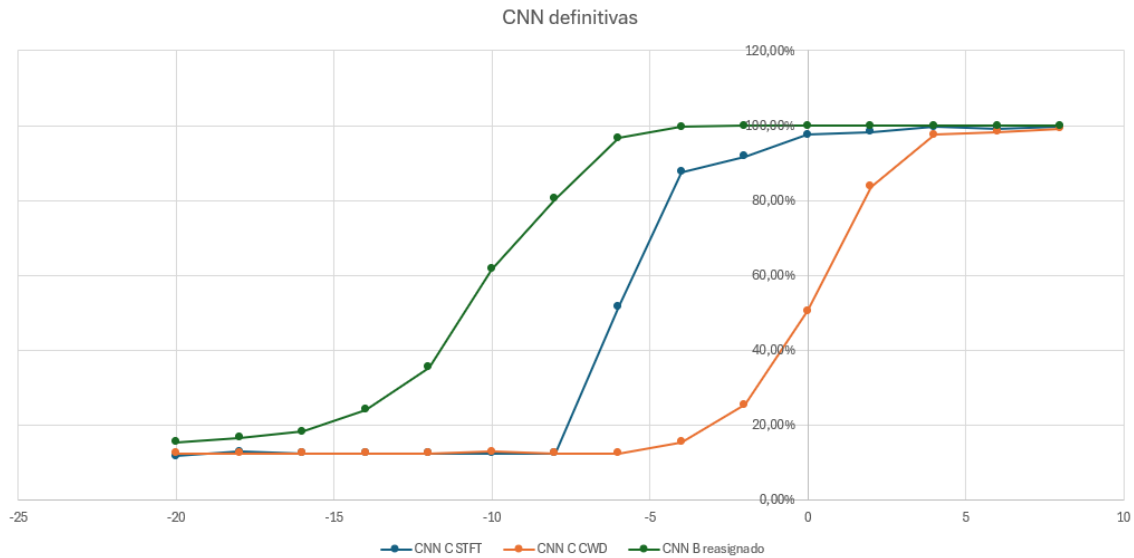


Figura 4-24. Resumen de resultados pruebas SNR de las CNN definitivas

El modelo CNN-C entrenado con espectrograma STFT (gráfica azul de la Figura 4-24) consigue precisiones de más del 90 % hasta los -2 dB de SNR, cayendo por debajo del 60 % por debajo de -6 dB. Su media de precisión dentro del intervalo de SNR en el que ha sido entrenado es del 90,62 %, siendo un resultado aceptable teniendo en cuenta el 94 % que consiguió la red en el test tras el entrenamiento. Por debajo de -6 dB la precisión baja a 12,5 %, o un octavo, lo que demuestra que la red no es capaz de diferenciar la señal del ruido y da como resultado la misma modulación para todas las predicciones. Es decir, fuera de su intervalo de entrenamiento, la red hace predicciones aleatorias, como demuestra su matriz de confusión a -10 dB (Figura 4-25).

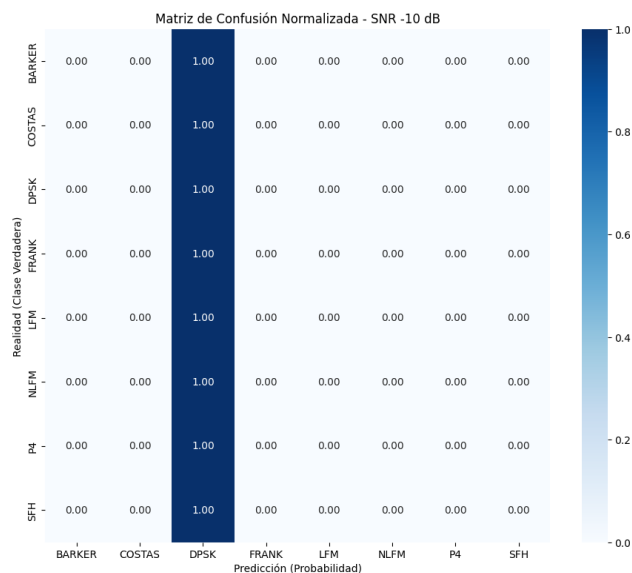


Figura 4-25. Matriz de confusión CNN-C en STFT a -10 dB

El modelo CNN-C entrenado en CWD (gráfica naranja de la Figura 4-24) tiene precisiones por encima del 80 % hasta 2 dB de SNR, cayendo por debajo del 60 % cuando la relación señal a ruido se reduce más de 0 dB. Su media de precisión dentro del intervalo de entrenamiento es del 60,30 %, volviendo a ser el modelo con peor precisión de los tres y estando muy por debajo del 92,25 % que consiguió en el test tras el entrenamiento. A partir de -2 dB la precisión se hunde por debajo del 40 %, empezando a trabajar con predicciones aleatorias desde -6 dB (Figura 4-26).

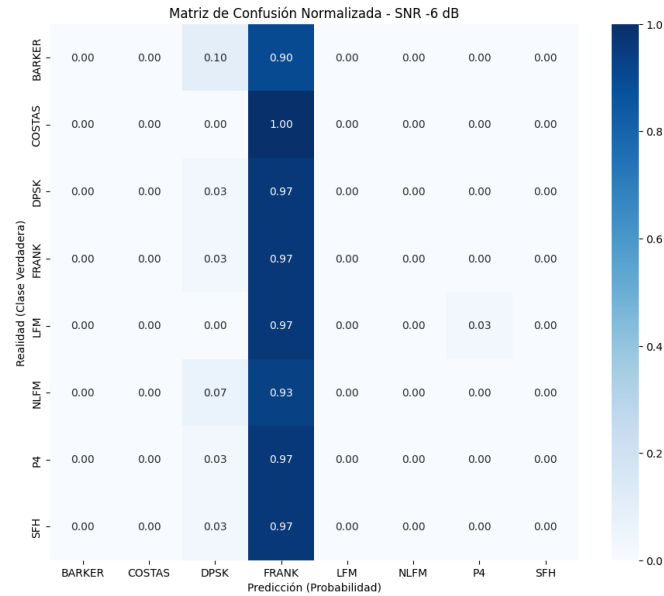


Figura 4-26. Matriz de confusión CNN-C en CWD a -6 dB

Sin embargo, la CNN-B entrenada en STFT reasignado ha demostrado una gran precisión en todo el intervalo de entrenamiento (>96,67 %), con una media del 99,53 %, que es prácticamente igual que lo conseguido en el test tras el entrenamiento. Cuando la relación señal a ruido cae por debajo de -6 dB, la red mantiene más de un 60 % de acierto hasta los -10 dB. Estos resultados demuestran la capacidad del espectrograma STFT reasignado de concentrar la potencia en torno a lo que denomina “centro de gravedad”, consiguiendo de esta forma paliar los efectos del ruido haciendo más afilada la señal.

4.5 Pruebas con dos pulsos por imagen

Los tres modelos CNN definitivos (CNN C en STFT, CNN C en CWD y CNN B en STFT reasignado), entrenados con imágenes que contenían un solo pulso, se han sometido a una serie de pruebas con dos pulsos, a fin de evaluar la capacidad de discriminación de las redes sin haber sido entrenadas para esta tarea (recordemos que el objetivo inicial era el de la clasificación de señales intrapulso). La base de datos de pruebas contenía 1200 imágenes (150 por modulación) por cada PRF elegido² (8 kHz, 12 kHz, 16 kHz y 20 kHz).

² Los PRF han sido elegidos en función de la información de [25]

PRF	Tamaño de ventana
8 kHz	150 μ s
12 kHz	130 μ s
16 kHz	90 μ s
20 kHz	75 μ s

Tabla 4-10. Relación PRF y tiempo de ventana de recorte

El principal problema del PRF para la red reside en la variación del tamaño de ventana. Mientras que para el entrenamiento se ha empleado una ventana de recorte de tiempo de 40 μ s (que permite mostrar el pulso completo con gran resolución), al tener que mostrar dos pulsos, esta ventana debe ampliarse (Tabla 4-10), con el fin de que el recorte de la imagen contenga ambos. Esto reduce la “escala” de la señal, al mantenerse fija tanto la resolución de la imagen como la resolución espectral, lo que dificulta la capacidad de la red de encontrar los patrones morfológicos aprendidos y discriminar entre modulaciones.

Los resultados de los 3 modelos CNN a las pruebas con varios pulsos se muestran en la Figura 4-27.

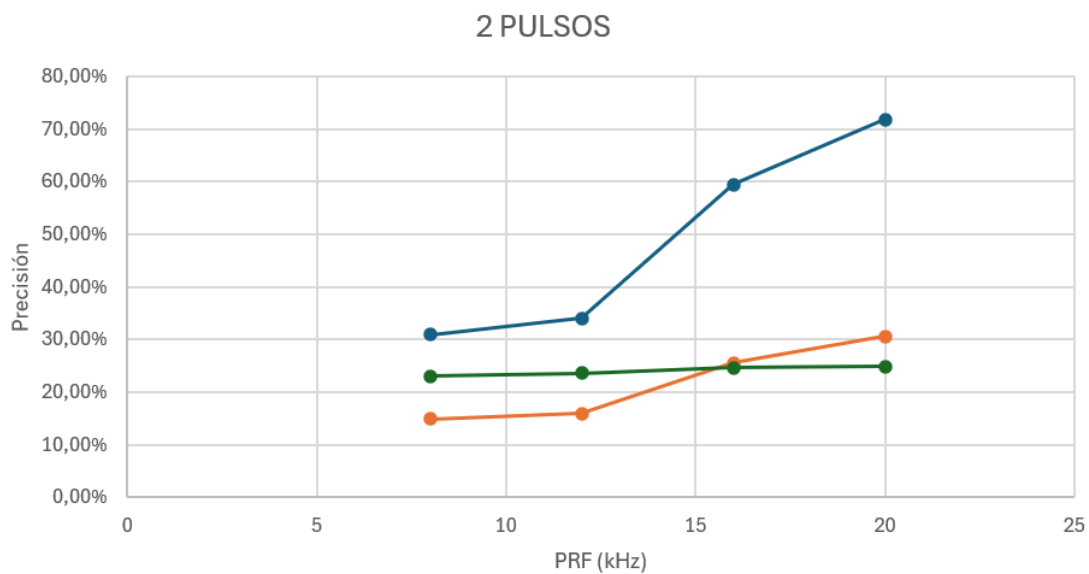


Figura 4-27. Resumen de resultados pruebas con 2 pulsos de las CNN definitivas

La CNN-C en STFT, ha conseguido los mejores resultados, con una precisión del 71,82 % para un PRF de 20 kHz. La precisión se mantendría por encima del 60 % hasta los 16 kHz, bajando rápidamente hasta el 34 % de acierto para un PRF de 12 kHz.

El análisis Grad-CAM (Figura 4-28) muestra que la red es capaz de distinguir ambas señales del ruido, lo que lleva a pensar que el problema está, como ya se ha comentado, en la resolución de cada pulso. Es decir, la red ve la señal, pero esta no contiene la información suficiente para discriminar entre modulaciones.

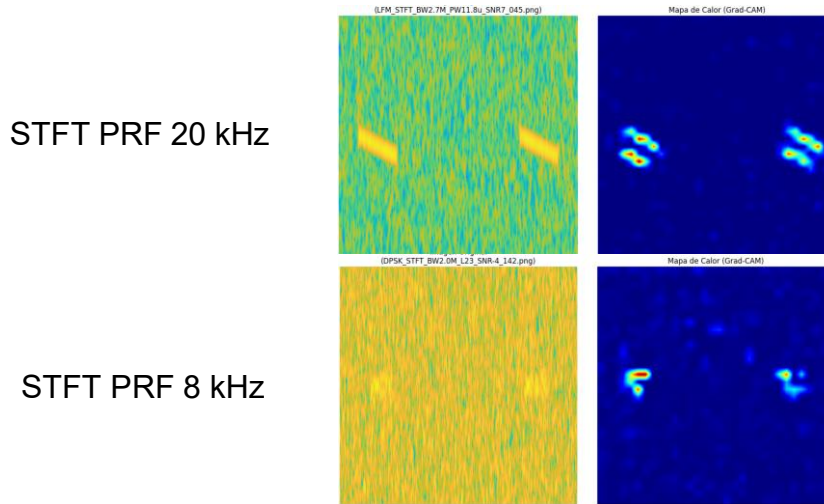


Figura 4-28. Análisis Grad-CAM de 2 pulsos en STFT

La CNN-C entrenada con la base de datos en CWD, ha conseguido una precisión del 30 % para un PRF de 20 kHz, que cae por debajo del 30 % cuando el PRF se reduce a 12 kHz y 8 kHz. El análisis Grad-CAM (Figura 4-29) muestra que la red también es capaz de distinguir los 2 pulsos de forma independiente, por lo que nuevamente el problema debe encontrarse en la falta de resolución. En el caso de la distribución de Choi-Williams, la red obtiene peor resultado que en STFT, por lo que puede ser que las propias características de la distribución hagan que la falta de resolución afecte en mayor medida al rendimiento.

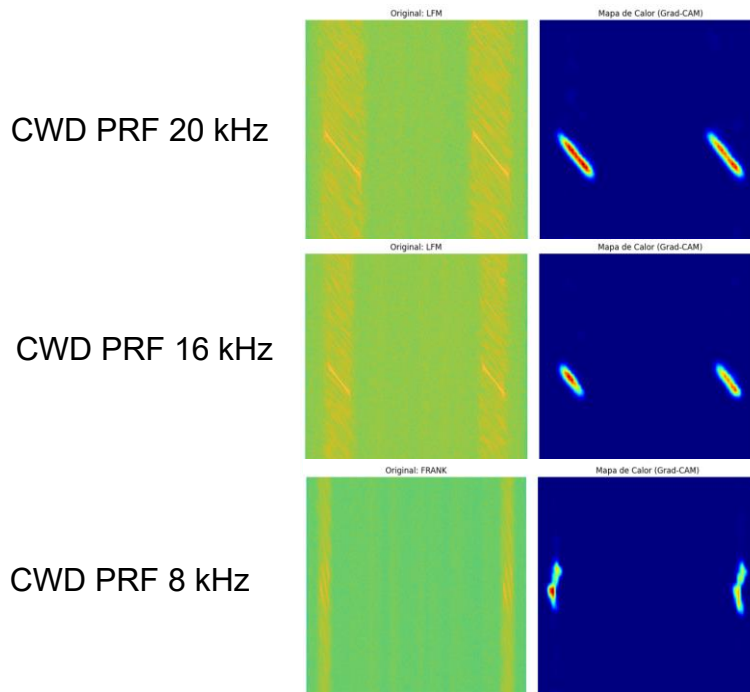


Figura 4-29. Análisis Grad-CAM de 2 pulsos en CWD

La CNN-B entrenada en STFT reasignado ha obtenido unos resultados malos pero estables. La red tiene una precisión del 24,83 % con un PRF de 20 kHz, manteniéndose por encima del 23 % a lo largo de toda la prueba, a diferencia del modelo en CWD, que cae hasta el 15 %. El análisis Grad-CAM (Figura 4-30) muestra una vez más que la red es capaz de distinguir los pulsos, si bien la precisión es más baja en STFT reasignado que en los otros dos modelos.

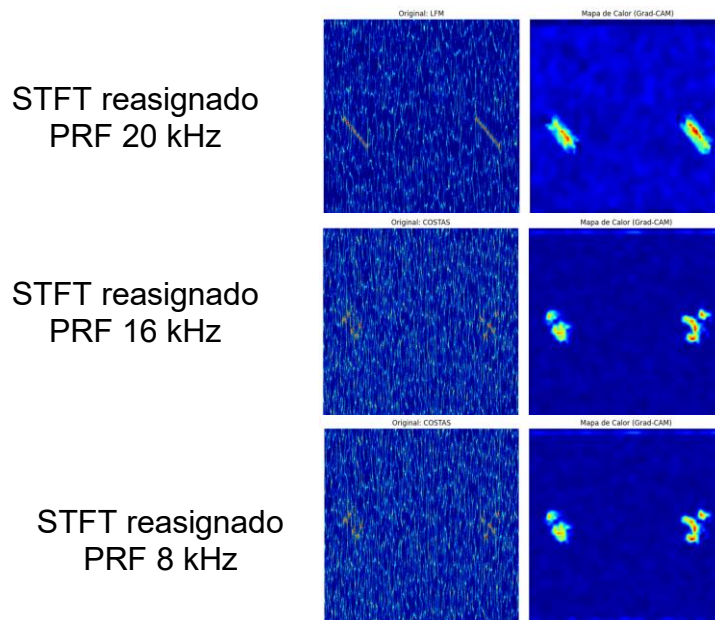


Figura 4-30. Análisis Grad-CAM de 2 pulsos en STFT reasignado

En conjunto, los resultados evidencian que ninguna de las tres arquitecturas evaluadas ha logrado un rendimiento satisfactorio en el escenario de dos pulsos, mostrando una limitada capacidad de generalización cuando las condiciones de entrada difieren significativamente de las del entrenamiento. La CNN-C basada en CWD presenta un fallo sistemático en todos los casos, lo que sugiere que, además de la pérdida de resolución, la propia arquitectura de la red neuronal no resulta adecuada para este tipo de representación. Por su parte, la CNN-B en STFT reasignado muestra un comportamiento estable pero claramente insuficiente, donde el afinado de la señal, si bien mejora la localización tiempo-frecuencia, también parece reducir la información disponible para la clasificación. No obstante, la CNN-C en STFT, aunque obtiene los mejores resultados relativos, sigue viéndose limitada por la disminución de resolución al ampliar la ventana temporal. En definitiva, estos resultados apuntan a que, con una resolución adecuada que preserve las características morfológicas de cada pulso, redes entrenadas con ejemplos de un solo pulso podrían potencialmente extender su capacidad de discriminación a escenarios con múltiples pulsos, si bien obtienen su mayor potencial en discriminación intrapulso.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Valoración de resultados

A lo largo del trabajo, se han desarrollado y evaluado diferentes arquitecturas de redes neuronales convolucionales (CNN-A, CNN-B y CNN-C) entrenadas sobre tres representaciones tiempo-frecuencia (STFT, CWD y STFT reasignado), así como su comportamiento frente a distintas condiciones de prueba.

En primer lugar, la CNN-A ha constituido un modelo base que permite validar la viabilidad del enfoque. Sus resultados muestran una capacidad inicial de clasificación, aunque con limitaciones tanto en precisión como en complejidad y capacidad de generalización, lo que ha llevado a la optimización y refinamiento del modelo.

La CNN-B introduce mejoras en la arquitectura y el proceso de entrenamiento, obteniendo un incremento notable en el rendimiento, especialmente en la representación STFT. En este caso, se observa que la red es capaz de extraer patrones característicos de las modulaciones con mayor fiabilidad. Sin embargo, las pruebas de robustez frente a reducción de SNR evidencian una serie de errores cometidos en el entrenamiento con las bases de datos en STFT y CWD, haciendo que el comportamiento real de la red sea muy deficiente al emplear estos modelos.

Por su parte, la CNN-C (que posee una arquitectura neuronal exactamente igual a la CNN-B, pero ha sido entrenada con otras bases de datos), tras la corrección de errores en la generación de la base de datos y el reentrenamiento, alcanza los mejores resultados globales del trabajo. La mejora es especialmente significativa en STFT, donde se obtiene una mejora considerable de la capacidad de generalización de la red.

Los resultados del entrenamiento en CWD no han cumplido las expectativas, obteniéndose unas tasas de acierto suficientes en entornos de bajo ruido, pero que se reducían notablemente en cuanto las condiciones no eran favorables para la red. Esto lleva a plantear si la forma de diseñar esta base de datos específica o la arquitectura CNN elegida le eran realmente favorables, o si, por consiguiente, se podría optimizar su rendimiento para estar a la altura de las representaciones STFT y de los resultados de [16] y [17].

Las 3 redes neuronales definitivas han superado el 90 % de precisión en condiciones favorables, mientras que la CNN-B entrenada con la base de datos en STFT reasignado ha demostrado una capacidad de generalización aceptable en entornos de elevado ruido. Esto lleva inevitablemente a plantear la validación, empleo y refinamiento de esta representación espectral para su aplicación en procedimientos de *Deep Learning*, que permitan mejorar las capacidades de identificación de señales de la Armada.

A su vez, las pruebas con dos pulsos por imagen muestran una pérdida generalizada de rendimiento en los tres modelos, evidenciando las limitaciones de las redes cuando se enfrentan a escenarios distintos a los utilizados durante el entrenamiento, particularmente en lo relativo a la resolución temporal y a la escala de la señal dentro de la imagen.

5.2 Líneas futuras

Basándonos en los resultados obtenidos en el transcurso de este trabajo de fin de grado, se proponen las siguientes líneas de investigación y mejoras para seguir estudiando la viabilidad del empleo de mecanismos de IA en el campo de la Guerra Electrónica:

5.2.1 Empleo de Transfer Learning

Se propone la implementación de técnicas de *transfer learning* mediante el uso de arquitecturas de redes neuronales convolucionales ya consolidadas, tales como ResNet, AlexNet o similares. Estas arquitecturas han demostrado un rendimiento sobresaliente en tareas de visión artificial y permitirían aprovechar los modelos previamente entrenados sobre las bases de datos de *Matlab*, reduciendo tanto el tiempo de entrenamiento como los requisitos de datos.

Esta aproximación permitiría mejorar la capacidad de generalización del modelo, así como incrementar la precisión en escenarios complejos, especialmente en condiciones de bajo SNR o con modulaciones de alta similitud, lo que aumentaría el rendimiento general de este método.

5.2.2 Optimización de las CNN diseñadas y pruebas en tiempo real

Otra línea de investigación relevante consiste en la adaptación específica de las arquitecturas de redes neuronales al tipo de espectrograma empleado (STFT, CWD o STFT reasignado), con el objetivo de optimizar su rendimiento. Esto implica modificar parámetros como el tamaño de entrada, el número de filtros, la profundidad de la red o las funciones de activación, de forma que se ajusten a las características particulares de las representaciones tiempo-frecuencia utilizadas.

Asimismo, se plantea como mejora fundamental el aumento del tamaño y la variabilidad del conjunto de datos. Aunque en este trabajo se ha empleado una base de datos sintética de 64,000 imágenes para entrenar cada red, la incorporación de una mayor diversidad de parámetros radar, condiciones de ruido y combinaciones de señales permitiría reforzar la robustez del modelo.

Finalmente, se propone probar la red con emisiones reales, lo que permitiría evaluar su comportamiento en entornos operativos y detectar posibles desviaciones respecto a los datos simulados. Además, la realización de las pruebas con emisiones en tiempo real permitiría evaluar la velocidad de la red en la realización de predicciones. Esta fase de test con señales reales constituye un paso clave para la futura implementación práctica del sistema en aplicaciones de guerra electrónica.

5.2.3 Empleo combinado de SNN o SVM junto con las CNN desarrolladas

Como evolución avanzada del modelo propuesto, se plantea la combinación de redes neuronales convolucionales con otras técnicas de inteligencia artificial, en particular las *Spiking Neural Networks* (SNN) y los clasificadores basados en *Support Vector Machines* (SVM).

Las SNN, inspiradas en el funcionamiento biológico del cerebro, presentan una alta eficiencia en el procesamiento temporal de señales, lo que las convierte en candidatas idóneas para el análisis de características dinámicas de emisiones radar. Su integración permitiría capturar información temporal no explotada en este trabajo como es el PRF, que aporta información clave para la identificación de emisores [29].

Por otro lado, el uso de SVM como clasificador final sobre características previamente extraídas por las redes neuronales permitiría mejorar la capacidad de separación entre clases, especialmente en casos donde las fronteras de decisión sean complejas, como se ha demostrado en [17].

La combinación de estas técnicas daría lugar a un sistema híbrido capaz no solo de clasificar modulaciones radar, sino también de avanzar hacia la identificación específica de emisores mediante la explotación conjunta de múltiples características de la señal. Este enfoque representa un paso significativo hacia sistemas más avanzados de inteligencia electrónica, con mayor precisión y capacidad de discriminación.

5.2.4 Estudiar el efecto de la resolución de imagen

Resulta de interés estudiar cómo la resolución de las imágenes generadas a partir de los espectrogramas influye en la capacidad del modelo para extraer características. Existe un compromiso entre la cantidad de detalle espectral capturado y la eficiencia computacional: resoluciones elevadas permiten preservar estructuras finas de la señal, pero pueden introducir redundancia y aumentar el coste de procesamiento, mientras que resoluciones reducidas pueden provocar la pérdida de información relevante para la clasificación.

Este análisis podría abordarse generando múltiples versiones del conjunto de datos con diferentes tamaños de imagen (y resolución espectral), manteniendo constantes el modelo de clasificación y las condiciones de entrenamiento, con el fin de aislar el efecto de la resolución. Se espera observar la existencia de un umbral inferior de tamaño a partir del cual el rendimiento se degrada de forma significativa, así como una región de saturación en la que incrementos adicionales de resolución no se traducen en mejoras apreciables en métricas como la precisión. Este comportamiento permitiría identificar un rango óptimo de resolución que maximice el rendimiento manteniendo un coste computacional contenido.

6 BIBLIOGRAFÍA

- [1] AJEMA, «Ministerio de defensa,» 2024. [En línea]. Available: <https://armada.defensa.gob.es/html/home/ARMADA2050.pdf>. [Último acceso: 20 01 2026].
- [2] F. S. Bergés, «Revista Ejércitos,» 2021. [En línea]. Available: <https://www.revistaejercitos.com/articulos/introduccion-a-la-guerra-electronica-naval/>. [Último acceso: 20 01 2026].
- [3] M. Skolnik, Radar Handbook, Tata McGraw-Hill company, 2008.
- [4] S. d. O. y. P. T. (SOPT), La Guerra Electrónica en España, Ministerio de Defensa, 2009.
- [5] M. A. Richards, J. A. Scheer y W. A. Holm, Principles of Modern Radar, SciTech Publishing, 2010.
- [6] G. Jin, Y. Deng, R. Wang, W. Wang, P. Wang, Y. Long, Z. Zhang y Y. Zhang, «An Advanced Nonlinear Frequency Modulation Waveform for Radar Imaging With Low Sidelobe,» *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, nº 9, pp. 6155-6168, 2019.
- [7] M. A. Richards, Fundamentals of Radar Signal Processing, McGraw-Hill Education, 2014.
- [8] C. Wolff, Radares y EW. Guía de bolsillo, Múnich: Rohde & Schwarz, 2004.
- [9] V. C. Chen y H. Ling, Time-Frequency Transforms for Radar Imaging and Signal Analysis, Norwood: Artech House, 2002.
- [10] F. A. Burbano, M. A. Ramos y E. F. Jojoa, «Análisis del desempeño de un sistema de comunicaciones con modulación DBPSK/DQPSK basado en hardware reconfigurable,» *Ing. USBMed*, vol. 5, nº 2, pp. 18-25, 2014.
- [11] M. d. Arcangelis, Historia de la Guerra Electrónica, Madrid: San Martín, 1983.
- [12] Comisión Europea, «Una definición de la Inteligencia Artificial: Principales capacidades y disciplinas científicas,» Grupo de Expertos de Alto Nivel sobre Inteligencia Artificial (AI HLEG), Bruselas, 2019.
- [13] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, MIT Press, 2017.

- [14] D. J. Matich, «Redes Neuronales: Conceptos Básicos y Aplicaciones,» Universidad Tecnológica Nacional. Facultad Regional Rosario , Rosario, 2001.
- [15] A. H. Rodríguez, «mlearninglab,» 2018. [En línea]. Available: <https://mlearninglab.com/2018/12/15/redes-convolucionales-de-la-invarianza-a-la-equivarianza-traslacional/>. [Último acceso: 21 01 2026].
- [16] Z. Luo, X. Wang y S. Yuan, «Radar Emitter Recognition,» *Multidisciplinary Digital Publishing Institute*, p. 22, 2024.
- [17] M. Zhang, M. Diao, L. Gao y L. Liu, «Neural Networks for Radar Waveform Recognition,» *Symmetry*, vol. 9, n° 5, p. 20, 2017.
- [18] Q. Guo, X. Yu y G. Ruan, «LPI Radar Waveform Recognition Based on Deep Convolutional Neural Network Transfer Learning,» *Symmetry*, vol. 11, n° 4, p. 14, 2019.
- [19] W. Elvira-García, «El espectrograma reasignado: creación, uso y utilidad en fonética,» de *53 reflexiones sobre aspectos de la fonética y otros temas de lingüística*, Universidad de Barcelona, 2022, pp. 355-361.
- [20] A. V. Oppenheim, R. W. Schaffer y J. R. Buck, *Discrete-time Signal Processing*, Prentice Hall, 1999.
- [21] A. G. Pérez, D. C. Martínez, M. V. Rodríguez y D. G. Lieberman, «APLICACIÓN DEL ESPECTROGRAMA MODIFICADO PARA LA IDENTIFICACIÓN DE MÚLTIPLES FALLOS COMBINADOS EN MOTORES DE INDUCCIÓN,» Tecnológico Nacional de México en Celaya, Celaya, 2022.
- [22] A. H. Jalalzadeh, H. Ebrahimi y M. J. Moghadam., «Classification of heart diseases using time-frequency representations of electrocardiogram signals by transfer learning networks.,» *Majlesi Journal of Electrical Engineering (MJEE)*, vol. 19, n° 1, pp. 1-8, 2025.
- [23] R. W. B. G. J. Mill, «Auditory-inspired interval statistic receivers for passive sonar signal detection,» de *OCEANS 2007 - Europe*, Sheffield, 2007.
- [24] S. A. Fulop y K. Fitz, «Algorithms for computing the time-corrected instantaneous frequency (reassigned) spectrogram, with applications,» *J. Acoust. Soc. Am.*, vol. 119, n° 1, pp. 360-371, 2006.
- [25] K. K. S. y N. E. M. Purabi Sharma, «Artificial Intelligence Aided Electronic Warfare Systems- Recent Trends and Evolving Applications,» *IEEE Access*, vol. 8, pp. 224761 - 224780, 2020.
- [26] IHS Jane's, *Jane's Radar and Electronic Warfare Systems*, Alexandria, VA: Jane's Information Group, 2010-2011.
- [27] Google, «Gemini,» [En línea]. Available: <https://gemini.google.com/app>. [Último acceso: 16 03 2026].
- [28] F. Auger, O. Lemoine, P. Gonçalves y P. Flandrin, «Time-Frequency Toolbox (TFTB),» s.f.. [En línea]. Available: <https://tftb.nongnu.org/>. [Último acceso: 20 02 2026].
- [29] G. Ben, X. Zheng, Y. Wang, X. Zhang y N. Zhang, «Chirp signal denoising based on convolution neural network,» de *2013 36th International Conference on Telecommunications and Signal Processing, TSP 2013*, 2013.

- [30] F. G. d. C. Martínez, «Identificación de emisiones radar mediante técnicas de inteligencia artificial,» 2022.
- [31] «Web de La Moncloa,» [En línea]. Available: <http://www.lamoncloa.gob.es>. [Último acceso: 13 enero 2015].
- [32] J. Rodríguez y V. Fernández, Cómo redactar el estado del arte de un trabajo, Editorial Genios, 2010.
- [33] P. Martínez y A. García, Cómo escribir una buena memoria de TFG, Publicaciones del 2000, 2013.
- [34] A. Pérez, Cómo escribir una bibliografía, Nuevas publicaciones.
- [35] S. J. Russell y P. Norvig, INTELIGENCIA ARTIFICIAL. UN ENFOQUE MODERNO, Madrid: PEARSON EDUCACIÓN, S.A., 2004.
- [36] IBM, «¿Qué son las redes neuronales convolucionales?,» 2024. [En línea]. Available: <https://www.ibm.com/es-es/think/topics/convolutional-neural-networks>. [Último acceso: 23 01 2026].
- [37] L. Cohen, Time-Frequency Analysis, Englewood Cliffs, NJ: Prentice Hall, 1995.
- [38] J. M. Ramírez Viáfara, M. M. Silva Zambrano, H. A. Romo Romero y P. E. Jojoa Gómez, «Detección espectral indirecta para un radar pulsante Wavelet,» *Scientia et Technica*, vol. 27, nº 1, p. 7, 2022.
- [39] A. M. Trobiani, «Clasificación de emisiones radar con redes neuronales,» Buenos Aires, 2008.
- [40] J. L. Jiménez Martín, «Nuevas técnicas de generación de señal para sistemas radar de alta resolución y baja probabilidad de interceptación,» Madrid, 2005.
- [41] Google, «NotebookLM,» [En línea]. Available: <https://notebooklm.google.com/>. [Último acceso: 16 03 2026].

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

El desarrollo y despliegue de sistemas basados en inteligencia artificial tiene importantes repercusiones sociales. Una de las cuestiones más relevantes es la dependencia de grandes volúmenes de datos etiquetados para entrenar modelos de aprendizaje automático, especialmente en el ámbito de la visión artificial.

Para crear estos conjuntos de datos es necesario realizar tareas de anotación manual de imágenes, como la clasificación de objetos, el marcado de regiones o la validación de etiquetas. Estas tareas suelen externalizarse a plataformas de trabajo digital, donde miles de trabajadores realizan pequeñas tareas repetitivas.

En los últimos años se ha documentado que parte de este trabajo se realiza en países con economías emergentes, donde los salarios son bajos y con regulaciones laborales más laxas. Un caso representativo es el de trabajadores en Madagascar, que realizan tareas de etiquetado de imágenes y preparación de datos para grandes empresas tecnológicas que desarrollan sistemas de inteligencia artificial. Estos trabajadores pueden pasar horas clasificando o segmentando imágenes para entrenar algoritmos de visión artificial, percibiendo remuneraciones muy reducidas en comparación con el valor económico generado por dichas tecnologías.

Este fenómeno plantea diversas cuestiones éticas y sociales:

- Precarización del trabajo digital, debido a la subcontratación de ciertas tareas y la ausencia de contratos laborales tradicionales.
- Desigualdad global en la cadena de valor tecnológica, donde el beneficio económico se concentra en grandes empresas mientras que el trabajo manual se externaliza a regiones con gran cantidad de mano de obra de baja cualificación.
- Falta de visibilidad del trabajo humano detrás de la inteligencia artificial, lo que genera la percepción errónea de que los sistemas funcionan de forma completamente autónoma.

Por otro lado, la inteligencia artificial está generando un impacto económico significativo en múltiples sectores industriales. Las tecnologías de visión artificial permiten automatizar tareas que anteriormente requerían intervención humana, como inspección visual, reconocimiento de objetos o análisis de imágenes.

Entre las principales implicaciones económicas destacan:

- Incremento de la productividad, al permitir automatizar procesos de análisis visual a gran escala.
- Reducción de costes operativos en sectores industriales, logísticos o sanitarios.
- Creación de nuevos mercados tecnológicos, relacionados con el desarrollo de modelos de inteligencia artificial, servicios en la nube y soluciones basadas en datos.

Sin embargo, también existen importantes efectos negativos:

- Desplazamiento de ciertos puestos de trabajo, especialmente aquellos relacionados con tareas repetitivas susceptibles de automatización.
- Concentración del mercado tecnológico en grandes empresas que poseen la infraestructura computacional y los recursos necesarios para entrenar modelos de gran escala.

En último lugar, el desarrollo de modelos avanzados de inteligencia artificial implica un alto consumo de recursos computacionales, lo que conlleva un impacto ambiental significativo.

El entrenamiento de modelos de aprendizaje profundo suele realizarse en centros de datos masivos, que albergan miles de servidores especializados. Estos centros de datos requieren grandes cantidades de energía eléctrica tanto para el funcionamiento del hardware como para los sistemas de refrigeración necesarios para mantener temperaturas adecuadas.

Entre los principales impactos ambientales asociados destacan:

- Elevado consumo energético, especialmente en el entrenamiento de modelos de gran tamaño.
- Emisiones indirectas de CO₂, dependiendo de la fuente de energía utilizada para alimentar los centros de datos.
- Consumo de agua en algunos sistemas de refrigeración utilizados en infraestructuras de computación a gran escala.

A medida que la inteligencia artificial continúa expandiéndose, el crecimiento de estas infraestructuras plantea desafíos relacionados con la sostenibilidad. En respuesta a ello, algunas empresas tecnológicas están comenzando a invertir en el uso de energías renovables en centros de datos, la mejora de la eficiencia energética del hardware o el desarrollo de modelos de IA más eficientes computacionalmente.

En definitiva, el desarrollo de sistemas de inteligencia artificial, particularmente en el ámbito de la visión artificial, presenta beneficios tecnológicos y económicos significativos, pero también plantea desafíos sociales y ambientales que deben ser considerados. Aspectos como el impacto energético de los centros de datos, la externalización del etiquetado de datos a trabajadores en economías vulnerables y las transformaciones del mercado laboral evidencian la necesidad de un desarrollo responsable y ético de la inteligencia artificial.

Por ello, además del avance técnico, resulta fundamental que el personal implicado mantenga una visión crítica sobre las implicaciones de las tecnologías que diseñan y desarrollan, integrando criterios de sostenibilidad, equidad social y responsabilidad económica en el proceso de innovación tecnológica.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

La inteligencia artificial está transformando de forma progresiva numerosos aspectos de la vida cotidiana. Los sistemas basados en aprendizaje automático se utilizan actualmente en ámbitos como la sanidad, el transporte, la industria, el comercio electrónico o los servicios digitales.

Desde una perspectiva ética, uno de los principales desafíos consiste en garantizar que el desarrollo de estas tecnologías se realice de manera responsable, transparente y equitativa. Los algoritmos de inteligencia artificial pueden influir en decisiones que afectan directamente a las personas, como la concesión de créditos, la selección de candidatos en procesos de contratación o la priorización de servicios públicos.

Asimismo, el creciente uso de la inteligencia artificial plantea cuestiones relacionadas con la privacidad y el tratamiento de datos personales, especialmente en aplicaciones basadas en reconocimiento de imágenes o análisis automatizado de información visual.

El avance de la inteligencia artificial está transformando ámbitos como el educativo. Herramientas basadas en IA permiten generar contenido educativo de mayor calidad más fácilmente, asistir en procesos de aprendizaje personalizados o automatizar ciertas tareas de evaluación.

Sin embargo, también surgen preocupaciones relacionadas con el uso indebido de estas herramientas. La facilidad para generar textos, resolver problemas o producir contenido automatizado puede fomentar prácticas de plagio académico, lo que plantea nuevos retos para los sistemas educativos.

Además, existe el riesgo de que una dependencia excesiva de estas herramientas reduzca el desarrollo de habilidades críticas en los estudiantes, como el pensamiento analítico, la creatividad o la capacidad de resolución autónoma de problemas. Por este motivo, resulta fundamental promover un uso responsable de la inteligencia artificial en la educación, donde estas tecnologías actúen como herramientas de apoyo y no como sustitutos del proceso de aprendizaje.

Otro de los debates éticos más relevantes en torno a la inteligencia artificial se relaciona con su posible utilización en el ámbito militar. Son muchos los países que llevan a cabo investigaciones para desarrollar sistemas autónomos capaces de realizar tareas de vigilancia, reconocimiento o incluso operaciones ofensivas.

En este contexto, el uso de inteligencia artificial en sistemas de armas autónomas plantea importantes cuestiones éticas y legales. La posibilidad de que sistemas automatizados puedan tomar decisiones que afecten a la vida humana genera un debate internacional sobre los límites que deben establecerse en el desarrollo de estas tecnologías, siendo los principales retos los siguientes:

- Delegación de decisiones críticas a sistemas automatizados, sin intervención humana directa.
- Dificultad en la atribución de responsabilidades en caso de fallos o daños causados por sistemas autónomos.

Debido a estas preocupaciones, diversas organizaciones internacionales, investigadores y entidades civiles han propuesto la necesidad de establecer marcos regulatorios y acuerdos internacionales que limiten o regulen el desarrollo de armas autónomas basadas en inteligencia artificial.

Desde el punto de vista social, las empresas tecnológicas están llevando a cabo desde los últimos veinte años la mayor acumulación de riqueza de la historia de la humanidad, creando una generación de oligarcas tecnológicos con capacidad de influir incluso en las elecciones de los líderes mundiales.

En definitiva, el desarrollo de sistemas de inteligencia artificial ofrece grandes oportunidades para el progreso tecnológico y social, pero también plantea desafíos éticos significativos. Su impacto en la

sociedad, la educación y el ámbito militar demuestra que estas tecnologías no son únicamente herramientas técnicas, sino que forman parte de un contexto social más amplio.

Por este motivo, los profesionales de la ingeniería deben ser conscientes de las implicaciones éticas de su trabajo y adoptar una actitud atenta y responsable en el desarrollo y aplicación de estas tecnologías.

ANEXO III: SIGLAS Y ACRÓNIMOS

AAN	Artificial Neural Network
AOA	Angle of Arrival
BPSK	Binary Phase-Shift Keying
BW	Band Width
CNN	Convolutional Neural Network
COMINT	Communication Intelligence
CW	Continuous Wave
DOA	Direction of Arrival
DPSK	Differential Phase-Shift Keying
ED	Exponential Distribution
ELINT	Electronic Intelligence
ENN	Elman Neural Network
EW	Electronic Warfare
FFT	Fast Fourier Transform
FMCW	Frequency Modulated Continuous Wave
FMOP	Frequency Modulation on Pulse
GE	Guerra Electrónica
IA	Inteligencia Artificial
IAG	Inteligencia Artificial Generativa
LFM	Linear Frequency Modulation
LPI	Low Probability of Interception
NLFM	Non-Linear Frequency Modulation
OBE	Orden de Batalla Electrónico
OTAN	Organización del Tratado del Atlántico Norte
PIRE	Potencia Isotrópica Radiada Equivalente
PMOP	Phase Modulation on Pulse
PRF	Pulse Repetition Frequency
PRI	Pulse Repetition Interval
PSL	Peak Side Lobe
PW	Pulse Width
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SEI	Specific Emitter Identification
SFH	Sequential Frequency Hopping
SIGINT	Signal Intelligence
SNN	Spiking Neural Network
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
SVM	Support Vector Machines
TDOA	Time Diference of Arrival
TOA	Time of Arrival
UE	Unión Europea

ANEXO IV: REPOSITORIO GITHUB

Los *scripts* empleados durante del desarrollo de este trabajo de fin de grado se encuentran disponibles en el repositorio: <https://github.com/manuelpelaezg/TFG43-Identificaci-n-autom-tica-de-emisores-radar.git>

Este repositorio está organizado en dos carpetas diferenciadas: “dataset_generation” y “neural_networks”.

La primera contiene los *scripts* de *Matlab* empleados para la generación y se ha dividido a su vez en 2 carpetas:

- “CNN-A y CNN-B”, que contiene los *scripts* empleados para generar las bases de datos que alimentan estos dos modelos neuronales.
- “CNN-C”, que contiene los *scripts* empleados para generar las bases de datos que se han usado para entrenar este modelo neuronal. Estos archivos acaban siempre en “_fix”, para evitar confundirlos.

Para cada espectrograma (STFT, CWD y STFT reasignado), se han creado 8 *scripts* (1 por modulación) y 1 *script* denominado maestro, que se emplea para ejecutar la generación de la base de datos.

En la carpeta “neural_networks”, están subidos los códigos “.ipynb” de *Python* en los que se han diseñado los distintos modelos neuronales. Se ha organizado la información en 3 carpetas (“CNN-A”, “CNN-B” y “CNN-C”), que contienen los archivos específicos para cada espectrograma usado.