Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Sistemas de la Telecomunicación

Trabajo de Fin de Grado

Estudio de la decodificación de señales DVB-S2 con ayuda de redes neuronales

Autor: Jorge Merino Puerta

Nombre tutor: Francisco Javier Escribano Aparicio

Curso 2020/2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Sistemas de la Telecomunicación

Trabajo de Fin de Grado

Estudio de la decodificación de señales DVB-S2 con ayuda de redes neuronales

Autor: Jorge Merino Puerta

Tutor: Francisco Javier Escribano Aparicio

Tribunal:

Presidente: D. Roberto Gil Pita

Vocal 1**:** D. Donato Rodríguez Alonso

Vocal 2**:** D. Francisco Javier Escribano Aparicio

Fecha: 6 de Octubre de 2021.

Índice

Lista de figuras

Lista de tablas

Resumen

Este trabajo trata de la implementación y entrenamiento de una red neuronal en un sistema DVB-S2 de vídeo, mediante una Demo de Matlab ya realizada, con el fin de poder decodificar los bits de los símbolos entrantes al sistema sin un gasto de recursos excesivo y sin tener grandes conocimientos acerca del ámbito.

Para ello se hablará tanto del sistema DVB-S2 como de las redes neuronales, los cuales serán los dos ”protagonistas” en este proyecto.

Por un lado se detallará parte por parte el sistema de transmisión y recepción DVB-S2 desde la entrada de información hasta la señal de salida del mismo sistema con el fin de contextualizar la ubicación de la red neuronal así como los datos de entrada que le llegaría en la implementación de la misma.

Por otro lado, se profundizará en conocer los fundamentos principales de las redes neuronales, los tipos de entrenamiento, y para finalizar los tipos de redes neuronales, con el propósito de contextualizar al lector para así definir en las Partes 5 y 6 las características de la red neuronal escogida para este trabajo, donde se plantean diversos experimentos para verificar la adecuación o no de su uso.

Palabras Clave

Red Neuronal

Entrenamiento

Capa Oculta

Modulación

Paquete

Constelación

Tasa de Código

Capa Física

Abstract

This work deals with the implementation and training of a neural network in a DVB-S2 video system, through a Matlab Demo that was done previously, in order to be able to decode the bits of the symbols entering the system without an excessive expenditure of resources and without having great knowledge about the field.

Due to all, we will talk about both the DVB-S2 system and neural networks, which will be the two ”protagonist” in this project.

On the one hand, DVB-S2 System will be explained from the input information to output signal in the same system in order to locate the neural network. The input data will arrive in the same system as well.

On the other hand, we we will prioritize in a deep knowledge on the main fundamentals of neural networks, the types of training, and to conclude the types of neural networks, in order to contextualize the reader in order to define in Parts 5 and 6 the characteristics of the Neural network chosen for this work, where various experiments are proposed to verify the adequacy or not of its use.

Keywords

Neural Network

Training

Hidden Layer

Modulation

Package

Constellation

Code Rate

Physical Layer

Glosario

Lista de acrónimos

DVB	*Digital Video Broadcasting*
LDPC	*Low Density Parity Check*
ACM	*Adaptative Codification and Modulation*
SDTV	*Standard Definition Television*
HDTV	*High Definition Television*
MPEG	*Moving Picture Experts Group*
FEC	*Forward Error Correction*
RS	Régimen de Símbolo
ATM	*Asynchronous Transfer Mode*
BCH	Bose-Chaudhuri-Hocquenqhem
BS	Servicios de Transmisión
FSS	*Fixed Satellite Service*
BSS	*Broadcast Satellite Service*
SI	Sistemas Interactivos
IRD	*Interactive Receiver Decoder*
CCM	*Constant Codification and Modulation*
VCM	*Variable Coding and Modulation*
UPL	*User Packet Length*
DFL	*Data Field Length*
MSB	*Most Significant Bit*
MS	*Mean Square*
QEF	*Quasi-Error-Free*
RF	Radio Frecuencia
TRA	Teoría de Resonancia Adaptada
BAM	*Bidirectional Adapted Memory*
IoT	*Internet of the Things*

RNA	Redes Neuronales Artificiales
VLSI	*Very Large Scale Integrated*
MLP	Perceptrón Multicapa
RBFN	*Radial Base Functions Networks*
SVM	*Support Vector Machine*
SOM	*Self-Organizing Maps*
RBP	Red *BackPropagation*
RMS	*Root Mean Square*
SNR	*Signal to NoiseRatio*
LLR	*Log Likelihood Ratios*
DTH	*Direct To Home*
DVB-RCS	*Digital Video Broadcasting - Return Channel System*
DVB-RCP	*Digital Video Broadcasting-Return Channel via Platforms*
IEEE	*Institute of Electrical and Electronics Engineers*
NCC	*Network Central Control*
RCST	*Return Channel Satellite Terminal*
DVB-RCG	*Digital Video Broadcasting - Return Channel GSM*

Capítulo 1

Introducción

1.1 Presentación

Los sistemas de comunicación por satélite han ido evolucionando de manera vertiginosa desde que se lanzó el *Telstar* en 1962 [12] y han tenido un papel vital en la evolución de la sociedad hacia una sociedad tecnológica. El punto principal de este trabajo y en lo que se va a centrar es en la problemática de la decodificación de las tramas de televisión digital del estándar DVB-S2. La mejor opción para decodificar las tramas de capa física de DVB-S2 es la basada en estimaciones probabilísticas, pero esto requiere de la utilización de muchos recursos en el receptor. En este sentido, las redes neuronales proporcionan una herramienta potencialmente útil para afrontar este problema.

En cuanto a las redes neuronales, se trata de unos sistemas de inteligencia artificial que se caracterizan por su gran flexibilidad y capacidad de adaptación, que les permite ser utilizados en numerosos escenarios.

Los casos de éxito mas conocidos son los casos de aplicaciones como *Google* o *Youtube*, ya que permiten que, a partir de los gustos del usuario, aparezcan recomendaciones de enlaces/vídeos en función del "aprendizaje" sobre la actividad de dicho usuario. Otros ejemplos son: el *dynamic pricing*, el sistema de ventas de *Amazon*, la identificación de riesgos en banca, la personalización de estrategias de marketing...

En definitiva, las redes neuronales están a la orden del día, integradas en las empresas más importantes de cada ámbito y con expectativas de que estas estas herramientas se puedan integrar en todo tipo de dispositivos electrónicos o apps.

1.2 Objetivos

Lo que se pretende con este proyecto es demostrar la viabilidad de implementación de una red neuronal a nivel de capa física donde cual su empleo es escaso, con el fin de racionalizar recursos. Para ello vamos a realizar pruebas utilizando como base de trabajo una demo de Matlab donde se va a implantar una red neuronal en el demodulador y en el codificador LPDC de un sistema DVB-S2 y se va a comprobar como de fiables son las redes neuronales a la hora de ayudar en la decodificación de señales, en este caso señales de vídeo que siguen el estándar DVB-S2.

Para llevar a cabo este proceso se entrenará a esta red neuronal y se llevará a estudio comparando la decodificación obtenida con el resultado matemático exacto y el aproximado (más costoso y mayor consumidor de recursos) y se analizará si es viable la implementación. Además se estudiará su comportamiento frente a distintas variaciones como el índice de modulación, tasa de codificación, distintas modulaciones, variación de la profundidad de la capa oculta de red... etc.

1.3 Estructura de la memoria

En los siguientes puntos se va a introducir un contexto teórico de los dos principales "personajes"de este trabajo.

Por un lado se empezará comentando el sistema DVB-S2 desde su contexto histórico, principales características y terminando por la estructura de este sistema y analizando a fondo que función tiene cada parte.

De igual manera, se comenzará con el otro "personaje"que son las redes neuronales, comentando las fechas señaladas en lo que ha descubrimientos de este ámbito se refiere, después se hablará de los tipos y el aprendizaje de éstas.

En el capítulo 4 se analizará la Demo de Matlab en la que se basa este trabajo, el cual está dividido en dos grandes partes.

La primera se basa en la comparación de las LLR Exacta, Aproximada y LLRNet con el fin de demostrar su implementación real a nivel de capa física y su rendimiento cuando se producen variaciones de: modulación, profundidad de las capas ocultas de la red y aumento/disminución del número de símbolos.

En la segunda se hallará la PER y de igual manera se analizará como varía si se producen variaciones en: número de símbolos y la profundidad de capa física.

El capítulo 5 se mostrarán los datos obtenidos en las simulaciones y se realizarán los análisis de los resultados.

Por último en el capítulo 6 estará dedicado a sacar las conclusiones del trabajo así como presentar las líneas futuras.

Capítulo 2

Descripción del estándar DVB-S2

2.1 Estudio y caracterización de tramas de capa física de señales DVB-S2

Para iniciar este trabajo se va a ahondar en cada parte del sistema que conforma el trabajo pero visto desde el punto de vista más teórico. Se profundizará primeramente, en dar la caracterización de un sistema/señal DVB-S2; seguidamente su codificación/decodificación, más adelante se profundizará en las partes del sistema DVB-S2.

Para entrar en contexto, el primer antecedente del sistema DVB-S2 es el DVB-S, fue creado en 1993 con el fin de transmitir señales de Televisión Digital y datos mediante enlaces satelitales para la banda *Ku* (intervalo del espectro electromagnético desde los 12-18 GHz) [24]. Este sistema es utilizado todavía por empresas.

Años más tarde, en 1997, se creó el DVB-DSNG (*Digital Satellite News Gathering*) "como una transmisión temporal u ocasional de noticias de televisión o audio para motivos de su radiodifusión, usando estaciones terrenas móviles con antenas transmisoras portátiles y transportables. Las principales diferencias destacables con el estándar DVB-S son la capacidad de uso de diferentes modulaciones digitales en la etapa de adaptación del satélite, lo que produce una reducción en el consumo de ancho de banda, ligado a un mayor consumo de potencia"[13].

En 2007, finalmente sale a la luz el sistema DVB-S2 donde su principal característica destacable es la flexibilidad en cuanto a sus propiedades en relación a que admite variabilidad en campos como la tasa de símbolo, ancho de banda del transpondedor o el código externo, así como una mayor gama de opciones en lo que a modulación o codificación frente a errores se refiere.

Tabla 2.1: Comparativa directa entre DVB-S y DVB-S2 [24]

.

Parámetro	**DVB-S**	**DVB-S2**
Codif. de vídeo	MPEG-2MP@ML	MPEG-2, MPEG-4
Codif. de Audio	MPEG-1 Capa 2	AAC, AC-3, MPEG-1
Multiplexación	MPEG Transport Stream	MPEG Transport Stream, ATM,IP.
Código externo	RS (204,188,8)	BCH variable
Código interno	Convol. (FEC: 1/2, 2/3, ¾, 5/6,7/8)	LDPC (FEC: 1/4, 1/3, 2/5, 3/5, 2/3)
Código interno		LDPC (FEC: 3/4, 4/5, 5/6, 8/9 y 9/10)
BW transpon.	Hasta 54 MHz	Variable
Roll-off	0.35 (coseno alzado)	0.2, 0.25, 0.35 (coseno alzado)
Modulación	QPSK, BPSK	QPSK, 8PSK, 16APSK, 32APSK
Tasa de símbolo	1-45 Ms/s	Variable

2.1.1 Descripción general del sistema DVB-S2

El sistema DVB-S2 pretende ir más allá y hacer un sistema de comunicaciones por satélite que permita:

- Radiodifusión de televisión SDTV y HDTV: es decir que tanto SDTV (a grandes rasgos televisión de definición estándar) como HDTV (televisión en alta calidad) pudiera llegar a la ciudadanía de manera masiva.
- Acceso a internet.
- Distribución de datos y servicios de Internet.

Los pilares en los que se basa DVB-S2 fijándose en la tabla anterior son:

- Permite un flujo de entrada variable, es decir, puede trabajar tanto con flujos de un solo paquete como con flujos continuos, incluyendo paquetes IP, ATM entre otros.
- En el código interno se puede observar que las tasas FEC *(Forward Error Correction)* alcanza desde 1/4 a 9/10 lo que permite una gran flexibilidad. El esquema FEC está caracterizado por usar códigos LDPC *(Low Density Parity Check) + BCH* que permite que opere cercano al límite de *Shannon*, es decir, casi sin errores.
- Amplio rango de modulaciones a elegir frente a DVB-S y más eficientes con la implantación de APSK.

- Un modo de trabajo llamad ACM (Codificación y Modulación Adaptativa), el cual permite optimizar cada trama o paquete con la modulación y codificación frente a errores óptima para transmitir el paquete.
- Tres señales distintas obtenidas al pasar dicha señal por un filtro de *Nyquist* con los tres valores de roll-off posibles (0.2, 0.25, 0.35).

Con estas características se consigue un aumento del 30 % de la capacidad del enlace en igualdad de condiciones de transmisión frente a DVB-S.

2.1.2 Escenarios de aplicación

Servicios de transmisión (BS):

DVB-S2 está dirigido a suministrar servicios DTH *(Direct To Home)* ya sea por FSS *(Fixed Satellite Service)* o por BSS *(Broadcast Satellite Service)*. Los servicios de Radiodifusión (BS) pueden ser transmitidos en formato MPEG, IP o ATM y se utiliza VCM (Modulación y Codificación Variable) como mecanismo de protección frente a errores y conseguir que los diferentes servicios trabajen de manera óptima (audio, vídeo, TV, HDTV) [6].

Existen dos modos en DVB-S2 para servicios de radiodifusión:

1. NBC-BS: servicios de transmisión no compatibles con otras versiones.
2. BC-BS: servicios de transmisión compatibles con versiones anteriores.

Cuando el número de receptores DVB-S2 sea considerablemente grande es probable que se requiera compatibilidad con versiones anteriores durante un período de tiempo, en el cual los receptores antiguos sigan captando la misma capacidad de radiodifusión mientras que, por otro lado, los nuevos receptores instalados de la versión nueva DVB-S2 reciban una capacidad adicional. Una vez que en todos los receptores se haya instalado el sistema DVB-S2 se podrá migrar a NBC-BS.

Servicios Interactivos (SI)

Uno de los objetivos de DVB-S2 es que los consumidores con IRD (Integrated Receiver Decoder) y a dispositivos personales. Como todo si debe tener un canal de retorno vía telefónica o satelital. DVB ofrece varios SI, por ejemplo:

- DVB-RCS [EN 301 790]: es un tipo de sistema satelital que fue estandarizado por la IEEE en el año 2000. Como se puede observar en la Figura 2.1, el sistema lo conforma un *Gateway*,

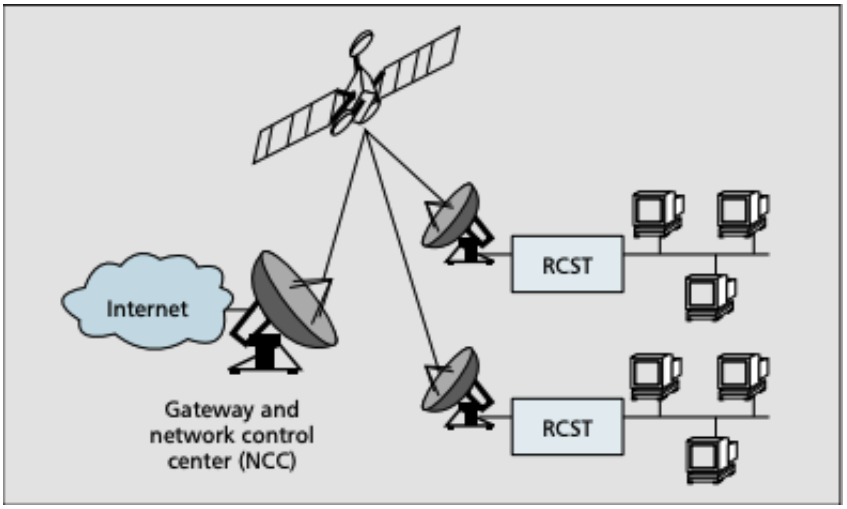

Figura 2.1: Sistema DVB-RCS. [2]

uno o más satélites que se encarguen de la distribución y transmisión de la información en ambos sentidos, una *Network Control Center* (NCC) responsable del seguimiento y del control y un *Return Channel Satellite Terminal* (RCST). Un aspecto importante para el éxito futuro de la multidifusión sobre los sistemas DVB-RCS es que garantiza que pueda implementarse de forma segura (por ejemplo, para evitar escuchas). Actualmente no hay ninguna disposición en la especificación DVB-RCS para asegurar multidifusión tráfico, y el objetivo de este sistema es proponer mecanismos que lo permitan. [2].

- DVB-RCP (ETS 300 801): basado en las infraestructuras *Public Switched Telecommunications Network* PSTN, "pueden soportar la implementación del RC para servicios interactivos adecuados para Sistemas DVB. PSTN se puede utilizar para implementar servicios interactivos en el entorno DVB, proporcionando un bidireccional vía de comunicación entre el terminal de usuario y el proveedor de servicios por medio de un módem. Para permitir el acceso a la PSTN, el terminal de usuario debe estar provisto de un módem (interno o externo al terminal de usuario). El módem constituye el módulo de interfaz de usuario para la interacción la red. El módem se conectará a la PSTN a través de la línea telefónica existente. Por lo tanto, compartirá línea con otros terminales / equipos ya presentes en las instalaciones del cliente (teléfonos, fax, otros, módems, etc.)"[5].

- DVB-RCG (EN 301 195): basado en la infraestructura GSM, "puede soportar la implementación del canal de interacción para sistemas de radiodifusión DVB, por proporcionar una ruta de comunicación bidireccional inalámbrica entre el terminal de usuario y una infraestructura que se conecta al proveedor de servicio"[22]. En la Figura 2.2 se puede observar la infraestructura GSM implementada.

  GSM es una tecnología de acceso inalámbrico que constituye la totalidad o una parte de la red de interacción. En GSM la red puede complementarse con otra red para llegar al

proveedor de servicios (comúnmente PSTN).

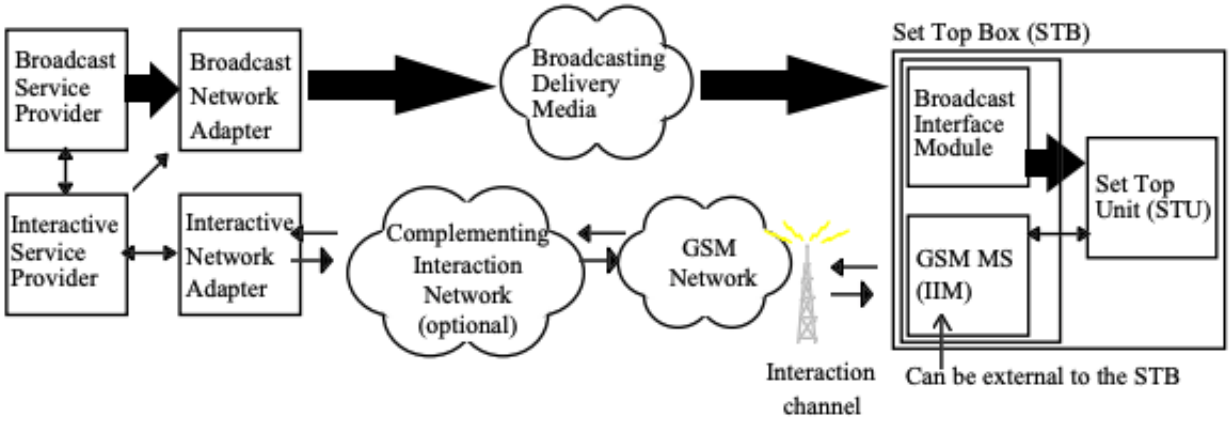

Figura 2.2: Sistema DVB-RCG [22].

.

El servicio de datos puede ser transportado con un formato de flujo de transporte o flujos genéricos:

- Flujos de Transporte (TS): es un método de transmisión a nivel 2 de MPEG utilizando paquetes TS. El paquete TS es de 188 bytes (longitud fija) de los cuales 4 son de encabezado y 184 son de datos, además también poseer un campo opcional de adaptación [10]. Esta cabecera tiene detalles de encriptación e información de marcas de tiempo para sincronizar un conjunto de Canales Lógicos TS relacionados y está subdividida en una serie de campos que se reflejan en la Figura 2.3:
  - Sync byte: campo de sincronización con respecto al resto de paquetes que conformen lo que se quiere enviar.
  - *Transport Error Indicator*: señala si el paquete contiene errores.
  - *Payload Unit Start Indicator*: señala si comienza un nuevo paquete.
  - *Transport Priority*: señala si el paquete debe tener prioridad frente a otros.
  - PID: es el indicador de cada paquete.
  - *Transport Scrambling Control*: señala si el campo de datos se encuentra encriptado o no.
  - *Adaptation Field Control*: señala si el paquete requiere del campo de adaptación opcional que se ha mencionado previamente.
  - *Continuity Counter*: contador que se incrementa con cada paquete con PID idéntico.

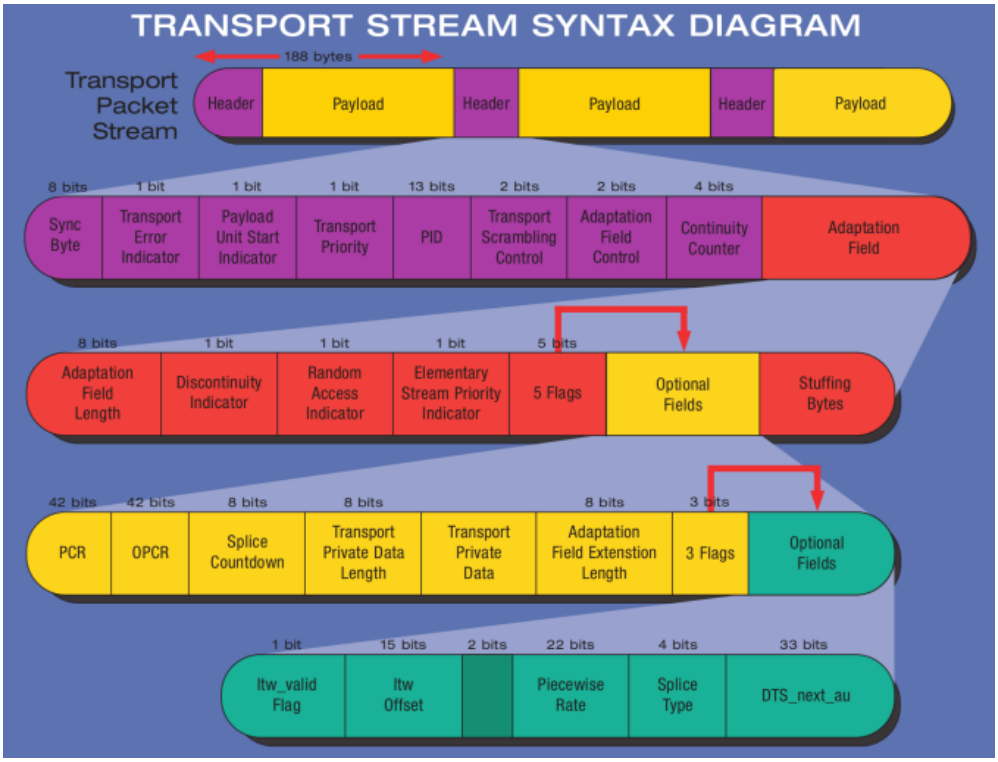

Figura 2.3: Estructura de un paquete TS [10].

- Flujos Genéricos: es un tipo de flujo de entrada en DVB-S2. Los flujos genéricos pueden trabajar en modo Paquetizado con paquetes de longitud fija o en modo Continuo que es un flujo de bytes sin ninguna estrctura específica o límites para el tamaño del flujo.

Contribución a la televisión digital y recopilación de noticias por satélite (DTV-C/DSNG)

Las aplicaciones de contribución de televisión digital por satélite consisten en comunicaciones punto-punto o punto-multipunto no están hechas para que llegue a la población general, para ello en los enlaces *uplink* se realizan conexiones con las estaciones receptoras que pueden ser fijas o transportables.

Siguiendo la Recomendación UIT-R SNG.770-1, se define DSNG como "Transmisión temporal y ocasional con poca antelación de televisión o sonido con fines de radiodifusión, utilizando estaciones terrenas de enlace ascendente altamente portátiles o transportables, etc" [6]. Por esta razón, los servicios de transporte trabajan de forma simple o múltiple en formato *MPEG 2 Transport Stream (TS)*.

Distribución de contenido de datos / *trunking* y otras aplicaciones profesionales (PS)

Las aplicaciones de Distribución de datos, incluso las PS, son mayoritariamente servicios punto-punto o punto-multipunto.

Los paquetes de datos pueden ser transportados de manera simple o múltiple a través de Flujos Genéricos.

Como mas tarde se mostrará DVB-S2 puede proveer de CCM *Coding and Constant Modulation*, VCM o ACM.

2.1.3 Arquitectura del sistema

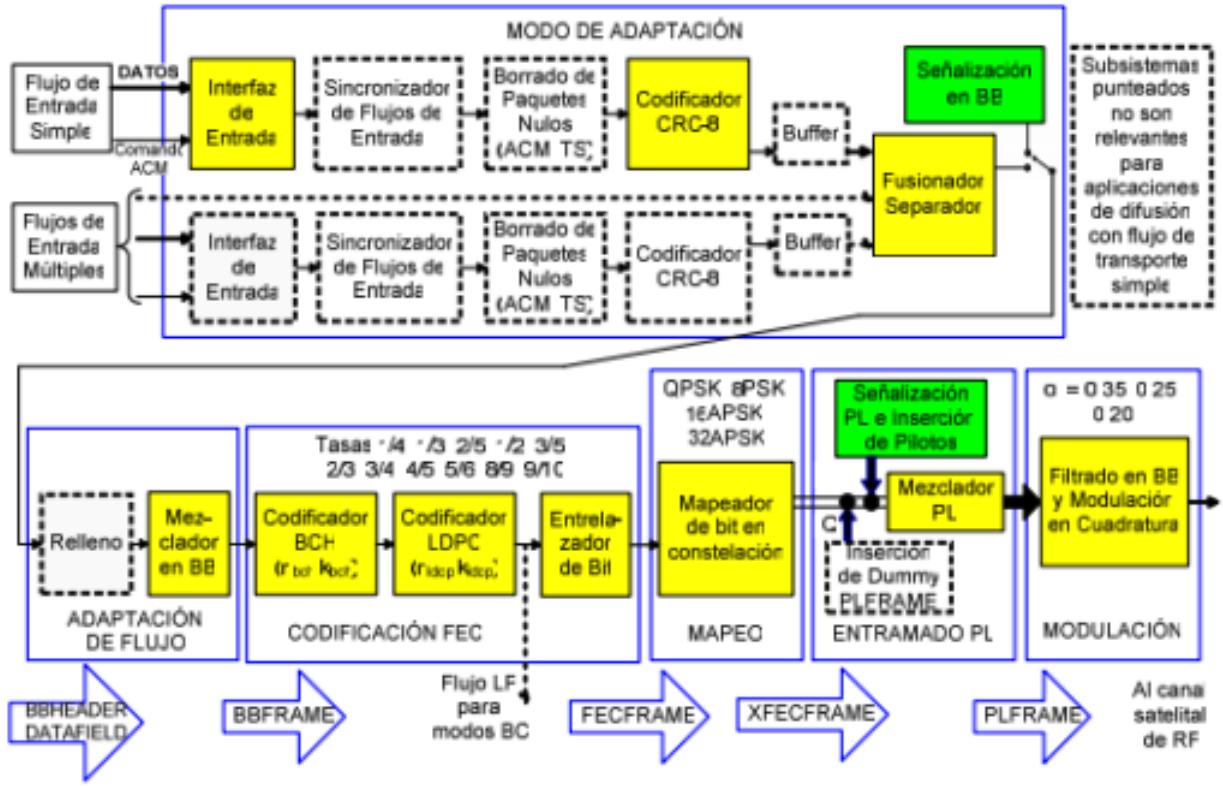

Figura 2.4: Arquitectura de un sistema DVB-S2 [8].

Las principales características de esta arquitectura que hacen que llegue a operar con una tasa de fallo cercana al límite de Shannon se debe al empleo del LDPC+BCH.

- Codificador de canal LDPC (*Low Density Parity Check*): "se basa en una matriz de codificación que establece la relación entre los bits de salida, o bits codificados, y los de entrada, o bits de información, esto es, define qué bits de información intervienen en la obtención de cada uno de los bits de salida a enviar por el canal de comunicación pertinente, en este

caso el espacio"[7]. Los parámetros de codificación LDPC se especifican en las Tablas 2.3 y 2.4 .

- Codificador BCH (*Bose-Chaudhuri-Hocquenqhem*): el codificador BCH aplica un código BCH que se puede ver en las tablas 2.3 y 2.4 en el que se protege fernte a posibles errores de hasta t bytes. Básicamente se podría decir que se utilizan para solucionar los errores que de manera esporádica tiene el codificador LDPC.

2.1.4 Especificaciones de los Subsistemas

Como ya se ha visto una de las características de DVB-S2 es la flexibilidad en la entrada de datos, permitiendo tanto flujos continuos de bits o flujos de paquetes de usuario con una longitud constante (Flujos Genéricos) como flujos de paquetes de usuario con una longitud constante de UPL= 188x8 bits (paquete MPEG) [6], dentro del cual el primer byte es el byte de Sync que ocupa 0x47h, como se ha explicado anteriormente.

Bloque "Modo de Adaptación"

Este bloque esta señalizado en la Figura 2.4 en la parte superior, y su funcionalidad se basa en:

1. Mediante la Interfaz de Entrada introduce los flujos de datos al sistema.

2. Seguidamente se realiza una sincronización de flujos, para poder garantizar unos retardos y unos regímenes binarios equitativos entre los flujos que se introduzcan (opcional).

3. Posteriormente se llega al *Null-Packet Deletion*, donde se borran los paquetes nulos MPEG, solo para Flujos de Transporte y para el ACM.

4. En el caso de Flujos múltiples se juntan en este punto todos los flujos.

5. Los flujos de entrada se dividen en *DATA FIELD* que es el campo donde se recogen los datos del paquete, se puede visualizar en la Figura 2.6.

6. Para acabar, se inserta señalización en banda base para notificar al receptor el Modo de Adaptación elegido. La secuencia de salida del Modo de Adaptación es un BBHEADER (80 bits) más un *DATA FIELD*.

a) Interfaz de entrada: la función de este subsistema es convertir es convertir el flujo de entrada a digital.

b) Sincronizador de flujo a la entrada: dado que el procesamiento de datos en el modulador produce retardos inevitables de duración variable, de lo que se encarga este subsistema es de proporcionar tasas de bits y retardos constantes para evitar desfases entre flujos de entrada que provoquen interferencias posteriormente.

c) Borrado de paquetes nulos: a veces llegan paquetes MPEG con contenido NULL a la interfaz de entrada con el fin de hacer constante una tasa de bits determinada. Para ACM y para Flujos de transporte estos paquetes nulos MPEG se eliminan de la trama de información para reducir la tasa de información y aumentar la protección de error en el modulador.

d) Codificador CRC-8: este codificador solo se utiliza para flujos genéricos Paquetizados, si la longitud del paquete entrante UPL es igual a 0, este codificador no actuará y dejará pasar el flujo entrante sin modificarlo. En el caso de que UPL sea distinto de 0, el paquete recogerá el byte de sync y el resto bytes de información que serán procesados por el codificador CRC de 8 bits. El polinomio generador es:

$$g(X) = (X^5 + X^4 + X^3 + X^2 + 1) * (X^2 + X + 1) * (X + 1) \tag{2.1}$$

En la Figura 2.2 se muestra el funcionamiento de este codificador.

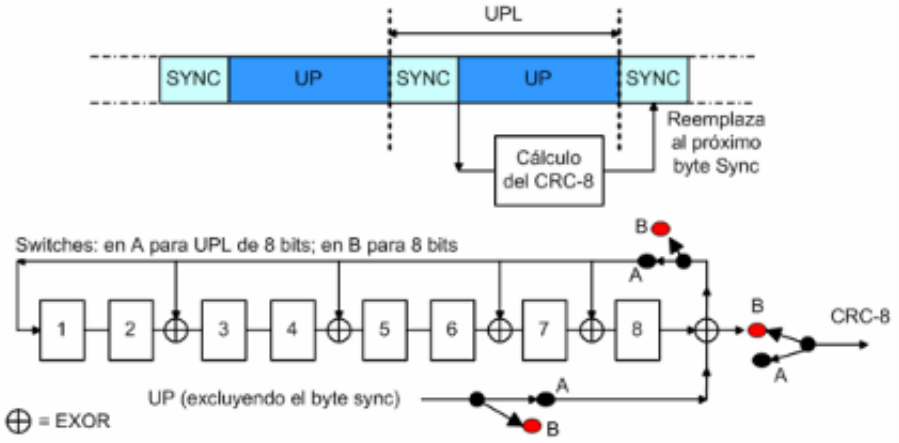

Figura 2.5: Codificación CRC-8. [9]

Como se muestra en la Figura 2.2 y como se ha explicado anteriormente, el cálculo de CRC-8 sustituye al byte SYNC del siguiente paquete.

e) Fusionador/Separador: según la Figura 2.4 el fusionador clasifica los flujos en Flujo Genérico Continuo o en Flujo Empaquetado y quedan a la espera almacenados hasta que el Fusionador/-Separador pueda procesarlos.

El Separador separa a su entrada, cada *DATA FIELD*, formado por DFL bits (Data Field Length) donde: Kbch –(10*8)>DFL>0.

Kbch es la longitud del bloque no codificado con BCH y a su vez depende de la longitud de la FRECFRAME y de la tasa de codificación que vendrá dada por los valores especificados en la Tabla 1.5a y 1.5b.

Un *DATA FIELD* está conformado por bits de un puerto de entrada y los cuales son transmitidos

con la misma codificación frente a errores y modulación. El fusionador finalmente reúne en una única salida los diferentes campos procesados

Si un *DATA FIELD* no está disponible en cualquier puerto de entrada del cuando el Fusionador/-Separador lo demande el Entramado de Capa Física genera y transmite un *Dummy* PLFRAME.

Cuando se sustituye el byte de sync por el CRC-8 es necesario que el receptor tenga algún método para recuperar la sincronización del paquete de Usuario.

Para ello, como se puede observar en la Figura 2.6 el Fusionador/Separador lee el número de bits que hay desde el comienzo del *DATA FIELD* hasta el comienzo del Paquete de Usuario, es decir el primer bit introducido por el CRC-8. Este dato lo almacena en SYNCD en Banda Base. Es decir, si el dato del campo SYNCD es 0, quiere decir que el primer paquete se encuentra alineado con el *DATA FIELD*.

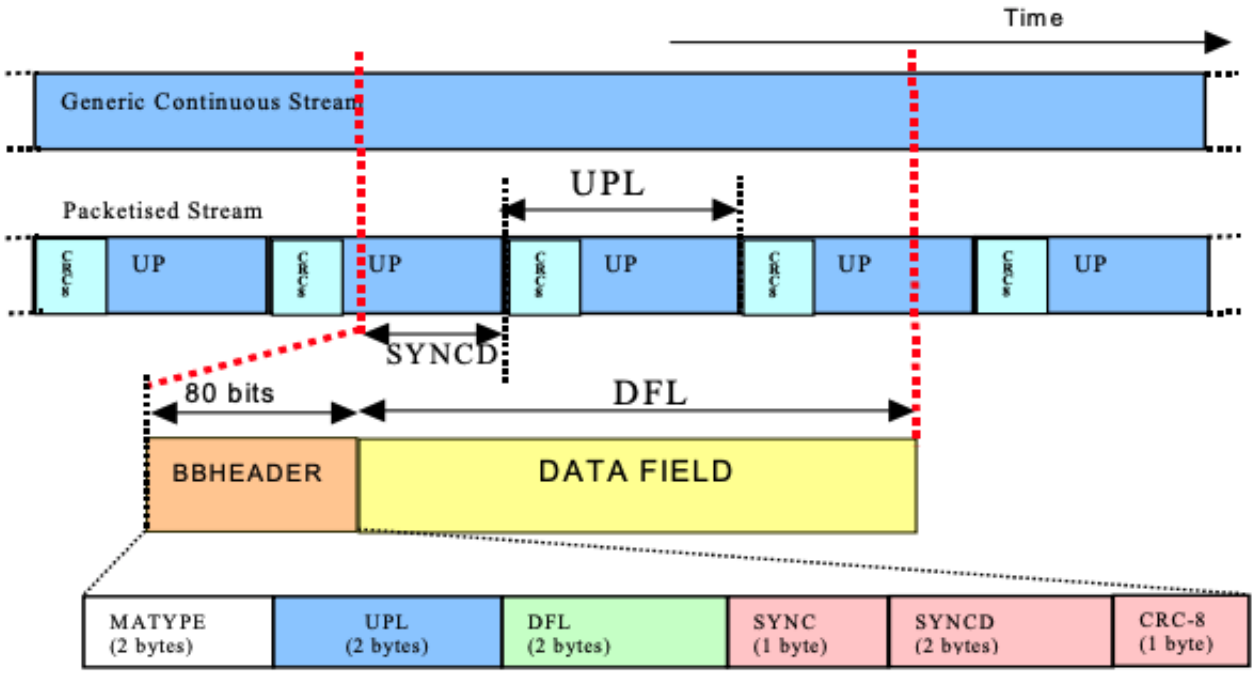

Figura 2.6: Funcionamiento Fusionador/Separador [6].

f) Inserción de la cabecera en Banda Base: como se puede observar en la Figura 2.4 anterior la cabecera BBHEADER, tiene una longitud fija de 80 bits (10bytes) y se inserta delante del *DATA FIELD*.

Esta cabecera está conformada por:

1. MATYPE (2 bytes): proporciona información acerca del tipo de Modo de Adaptación y el factor de *roll-off* como se especifica en la Tabla 2.2. Más a fondo el primer byte está compuesto por:

   (a) Campo TTS/GS (2 bits): precisa el tipo de flujo: Transporte o Genérico.

(b) Campo SIS/MIS (1bit): precisa si el flujo es de Entrada Simple o Múltiple.

(c) Campo CCM/ACM (1bit): precisa Modulación constante o Modulación y Codificación Adaptativa (CCM/ACM).

(d) ISSYI (1 bit): si este campo se encuentra a nivel alto, este campo se inserta detrás del Paquete de Usuario.

(e) NPD (1 bit): precisa la eliminación de paquetes nulos.

(f) RO (2 bits): precisa el factor de *roll-off*.

Tabla 2.2: Tabla de Valores MATYPE [6]

TS/GS	SIS/MIS	CCM/ACM	ISSYI	NPD	RO
11= Transporte	1= Simple	1=CCM	1= ON	1= ON	00=0.35
00= Gen. Empaq.	0= Múltiple	0= ACM	0= OFF	0= OFF	01= 0.25 10=0.2 11= R

Donde R es: Reservado. Si el SIS/MIS = 0, el byte MATYPE: Identificador de Flujo de Entrada.

Si el SIS/MIS= 1, este byte se reserva.

2. UPL (User Packet Length) (2 bytes): especifica la longitud del paquete de usuario en el rango [0, 65535].

3. DFL (Data Field Length) (2 bytes): especifica la longitud del DATA FIELD en el rango [0, 58112].

4. SYNC (1 byte): se trata de la réplica del byte sync del paquete de usuario.

5. SYNCD (2 bytes): número de bits comprendidos entre el comienzo del DATA FIELD hasta el primer bit del paquete de usuario.

6. CRC-8 (1 byte): código de detección de errores que se aplica a los primeros 9 bytes del BBHEADER.

Bloque "Adaptación de Flujo"

Este bloque se encarga de dar un tamaño constante (Kbch) a la BBFRAME de salida, completando las tramas en el caso que se requiera con relleno, así como un mezclado de la BBFRAME y está especificado en la Figura 2.4.

1. Relleno y Mezclador en Banda Base: Este relleno se emplea cuando los bits del DATA FIELD no completan la BBFRAME y es requisito que la BBFRAME tenga una longitud

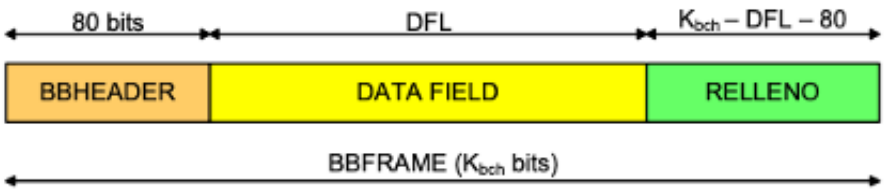

Figura 2.7: Trama BBFRAME [6]

constante como hemos visto anteriormente. Una vez se conforma la BBFRAME por completo esta es pasada por un proceso de aleatorización para proteger la información ante posibles errores, facilitando así la tarea del FEC que es el paso siguiente. En la Figura 2.7 queda especificado como queda estructurada la trama BBFRAME.

Bloque "Codificación FEC"

La codificación frente a errores se realiza concatenando el código externo BCH y los códigos internos LDPC. Estos bloques pueden tener una longitud variable, desde 16200 a 64800 bits. Seguidamente se aplica un entrelazado de bits a bits codificados FEC para 8PSK, 16PSK, 32APSK con el fin de separar bits mapeados en la misma señal de transmisión.

Cada BBHEADER se pasa por el codificador FEC lo que da lugar a una nueva trama llamada FECFRAME, que queda conformada de esta forma:

1. BCHFEC: bits de chequeo de paridad del código externo BCH.
2. LDPCFEC: bits de chequeo de paridad del codificador LDPC.

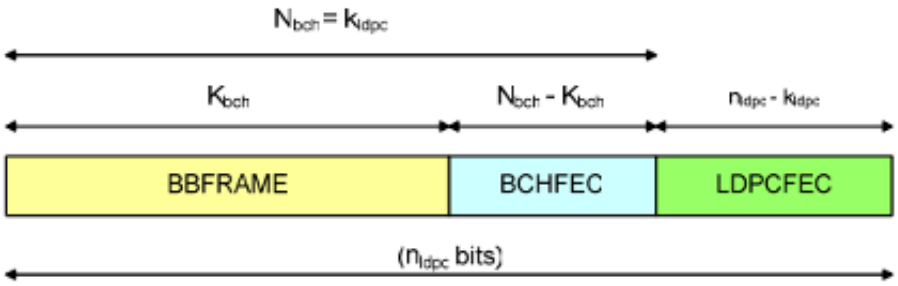

Figura 2.8: Trama FECFRAME [6]

1. Codificación externa BCH: siguiendo las tablas 2.3 y 2.4, se puede observar los parámetros de codificación BCH que le corresponde a cada BBFRAME a los cuales se aplica la codificación BCH para generar un FECFRAME, especiicada ne la Figura 2.8.
2. Codificación interna LDPC: de igual manera fijándose en las tablas 2.3 y 2.4 se podrá obtener la codificación LDPC para cada paquete.

Tabla 2.3: Parámetros de Codificación para Paquete de longitud normal [6]

Cód. LDPC	BNC BCH	BC BCH/BNC LDPC	Correc. t-error BCH	BCN LDPC
1/4	16008	16200	12	64800
1/3	21408	21600	12	64800
2/5	25728	25920	12	64800
1/2	32208	32400	12	64800
3/5	38688	38880	12	64800
2/3	43040	43200	10	64800
3/4	48408	48600	12	64800
4/5	51648	51840	12	64800
5/6	53840	54000	10	64800
8/9	57472	57600	8	64800
9/10	58192	58230	8	64800

3. Entrelazado de bit: es muy útil para evitar ráfagas de bits erróneos de manera consecutiva. En modulaciones 8PSK, 16APSK y 32APSK los bits que salen del codificador LDPC serán entrelazados utilizando un bloque "Entrelazador" con el fin de separar bits mapeados en la misma señal transmitida. Los datos se introducen en forma de columnas y en serie; sin embargo, los lee en forma de fila, también en serie.

Bloque "Mapeo"

De este modo se aplica ahora el mapeo a cada FECFRAME en la constelación, obteniendo un XFECFRAME compuesto por 64800/mu MOD o 16200/mu MOD, donde mu MOD es el número de bits transportados por un símbolo en una constelación.

El MSB de la FECFRAME se mapea como el MSB de la primera secuencia paralela. Estas secuencias paralelas son mapeadas en la constelación generando una secuencia nueva. Dependiendo de la eficiencia al modular se conseguirá una longitud u otra, ver Figura 2.6.

1. Mapeo en QPSK: emplea codificación Gray y sigue la tabla anterior para determinar los símbolos en la constelación. De esta forma, dos bits de la FECFRAME son mapeados en un símbolo QPSK.

2. Mapeo en 8PSK: también emplea codificación *Gray*, y el mapeo vendrá dado por el mapeo de bits viene dado por la Tabla 2.5. Donde 3 bits de la FECFRAME corresponderán a un símbolo en esta constelación.

3. Mapeo en 16APSK: como se puede observar en la Figura 2.12 esta constelación presenta

Tabla 2.4: Parámetros de Codificación para Paquete de longitud corta [6]

Cód. LDPC	BNC BCH	BC BCH/BNC LDPC	Error BCH	Tasa LDPC	BCN LDPC
1/4	3072	16200	12	1/5	16200
1/3	5232	21600	12	1/3	16200
2/5	6312	25920	12	2/5	16200
1/2	7032	32400	12	4/9	16200
3/5	9552	38880	12	3/5	16200
2/3	10632	43200	12	2/3	16200
3/4	11712	48600	12	11/15	16200
4/5	12432	51840	12	7/9	16200
5/6	13152	54000	12	37/45	16200
8/9	14232	57600	12	8/9	16200
9/10	NA	58230	12	NA	NA

Tabla 2.5: Estructura Entrelazador de Bits

Modulación	Filas trama larga	Filas trama corta	Columnas
8PSK	21600	5400	3
16APSK	16200	4050	4
32APSK	12960	3240	5

dos anillos concéntricos de radios R1, R2 y a su vez con 4 y 12 símbolos en cada anillo respectivamente. Donde Gamma será igual a la relación entre R1 y R2 como queda especificado en la Tabla 2.6. Además en la Tabla 2.8 queda reflejado la relación

4. Mapeo en 32APSK: como se puede observar en la Figura 2.13 presenta una constelación con 3 anillos concéntricos de radios R1, R2 ,R3 con 4, 12 y 16 símbolos en cada anillo respectivamente. En la Tabla 2.7 queda especificado la eficiencia espectral para cada tasa de código posible así como Gamma 1 y Gamma 2 que representa la relación óptima entre los Radios R1-R2 y R2-R3 respectivamente.

Bloque "Entramado de Capa Física"

Este subsistema se encarga de generar la trama física que recibe el nombre de PLFRAME de cada XFECFRAME y sigue una serie de pasos:

1. Inicialmente se genera la trama Dummy.
2. El XFECFRAME entrante se divide en slots, en concreto 90.

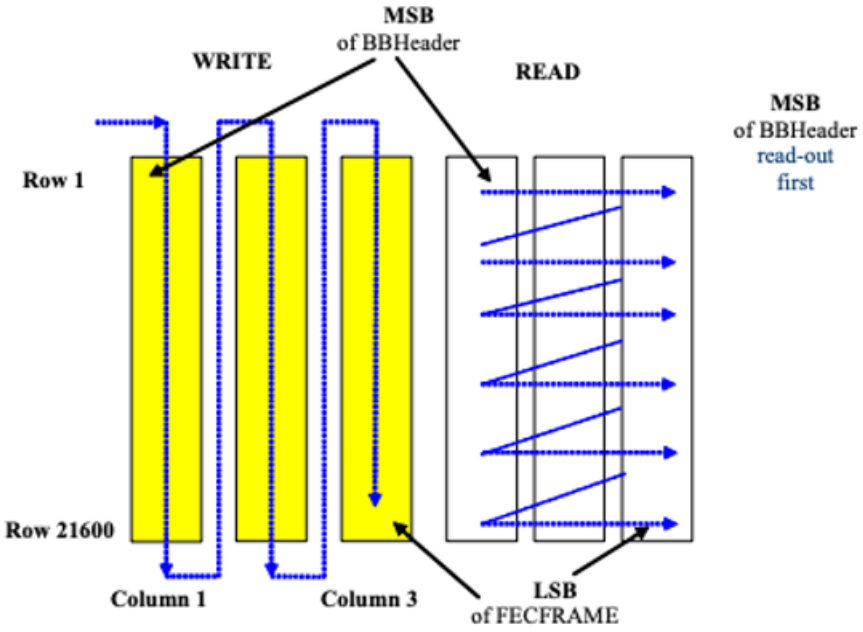

Figura 2.9: Trama FECFRAME [6]

Tabla 2.6: Relación de Radio de Constelación Óptima para 16APSK [6]

Tasa de Código	Eficiencia Espectral de Codificación	Gamma
2/3	2.66	3.15
3/4	2.99	2.85
4/5	3.19	2.75
5/6	3.32	2.70
8/9	3.55	2.6
9/10	3.59	2.57

3. Se inserta la cabecera llamada PLHEADER (que ocupa un slot adicional) delante del XFECFRAME y así se conforma el PLFRAME, como se puede observar en la Figura 2.14.

4. Se insertan los bloques pilotos, su papel es colaborar con la sincronización en el receptor y se colocan cada 16 slots.

5. Por último se procede a una aleatorización de los símbolos modulados por medio de un mezclador de capa física.

- Inserción de la trama *Dummy*: .[el] subsistema de entramado de capa física de DVB-S2 genera y transmite una trama *Dummy* cuando la XFECFRAME no esté lista para ser procesada y transmitida. Una *Dummy* PLFRAME se compone de un PLHEADER y de 36 slots de portadoras no moduladas"[8].

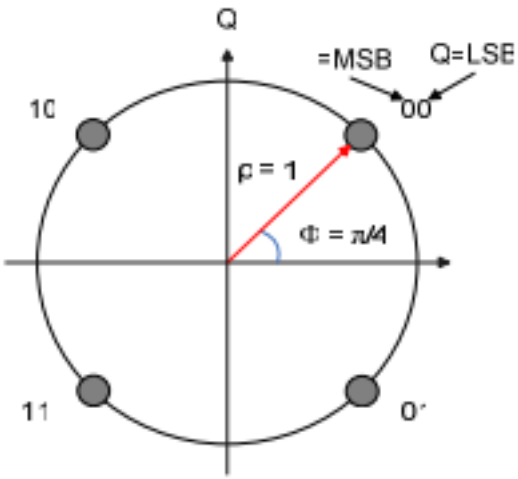

Figura 2.10: Mapeo QPSK [9].

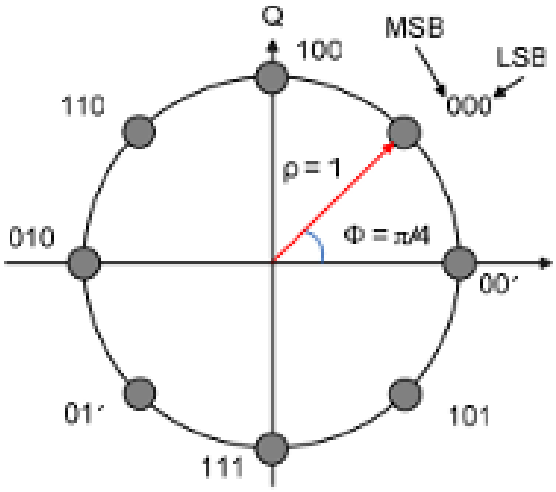

Figura 2.11: Mapeo 8PSK [9].

- Señalización de Capa Física: las razones fundamentales por las que se genera e introduce el PLHEADER es para proporcionar señalización de capa física y ayudar en la sincronización con el receptor. El PLHEADER está modulado en pi/2 BPSK y lo componen 90 símbolos. Una vez decodificados, el PLHEADER está compuesto por:
  - SOF (26 símbolos): indica inicio de trama.
  - Código PLS (64 símbolos): su función está orientada a la señalización de la capa física, y se compone de 7 símbolos subdivididos en:
    - MODCOD (5 símbolos): indica modulación y la tasa FEC.
    - TYPE (2 símbolos): indica la longitud de la FECFRAME. Informa sobre si una trama es larga (64800 bits) o corta (16200), así como la presencia de pilotos.
- Inserción de pilotos: la PLFRAME puede utilizar pilotos o no. En el caso de que tenga

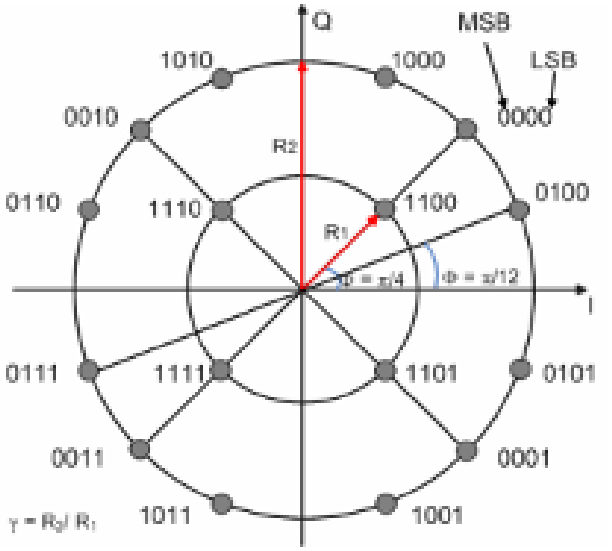

Figura 2.12: Mapeo 16APSK [9].

Tabla 2.7: Relación de Radio de Constelación Óptima para 32APSK [6]

Tasa de Código	Eficiencia Espectral de Codificación	Gamma1	Gamma2
3/4	3.74	2.84	5.27
4/5	3.99	2.72	4.87
5/6	4.15	2.64	4.64
8/9	4.43	2.54	4.33
9/10	4.49	2.53	4.30

pilotos, se inserta un BLOQUE PILOTO dentro de la PLFRAME, que será inmediatamente después del *slot* 16 y el siguiente será cada 32 slots y así sucesivamente con el fin de facilitar la sincronización confiable en el receptor sobre la estructura del bloque FEC. Este bloque piloto está compuesto por 36 símbolos (no modulados) y queda especificada en la Figura 2.15.

Si la posición del BLOQUE PILOTO coincide con el inicio del siguiente inicio de trama, entonces el BLOQUE PILOTO no se transmite.

- Mezclado de Capa Física: antes de modular cada PLFRAME entrante, se deberá aleatorizar esta trama menos la cabecera para conseguir que no haya picos de dispersión, para ello se multiplica cada PLFRAME por una secuencia de aleatorización compleja.

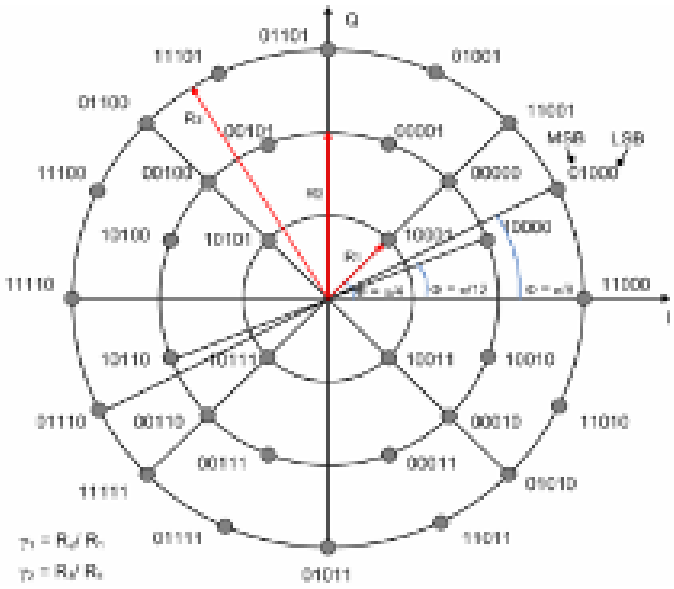

Figura 2.13: Mapeo 32APSK [9].

Tabla 2.8: Número de Slots por cada XFECFRAME [6]

	Trama Normal		Trama Corta	
mu mod [b/símb.]	**S**	**mu(%) sin pilotos**	**S**	**mu(%) sin pilotos**
2	360	99.72	90	98.90
3	240	99.59	60	98.36
4	180	99.45	45	97.83
5	144	99.31	36	97.30

Bloque "Mapeo"

Cada PLFRAME (mezclado) llega a este subsistema y pasa por un filtro de coseno alzado. Se podrá escoger entre tres valores de *roll-off*: 0.35, 0.25, 0.2 en función de las necesidades del servicio.

Seguidamente, tras filtrar la señal en banda base, se realiza la modulación en cuadratura, para generar la seña de Radio Frecuencia (RF).

- Rendimiento de DVB-S2 Frente a Errores: en la Figura 2.16 se puede observar los valores de Es/No (Energía Promedio por Símbolo Transmitido) para Tramas grandes (64800) y para tramas pequeñas (16200) para que el sistema trabaje en QEF.

Según se muestra en la tabla x estas Es/No están calculadas para tramas con una FECFRAME de longitud 64800.

Por otro lado el sistema DVB-S2 con FECFRAME corto se advierte en el estándar que se deberá

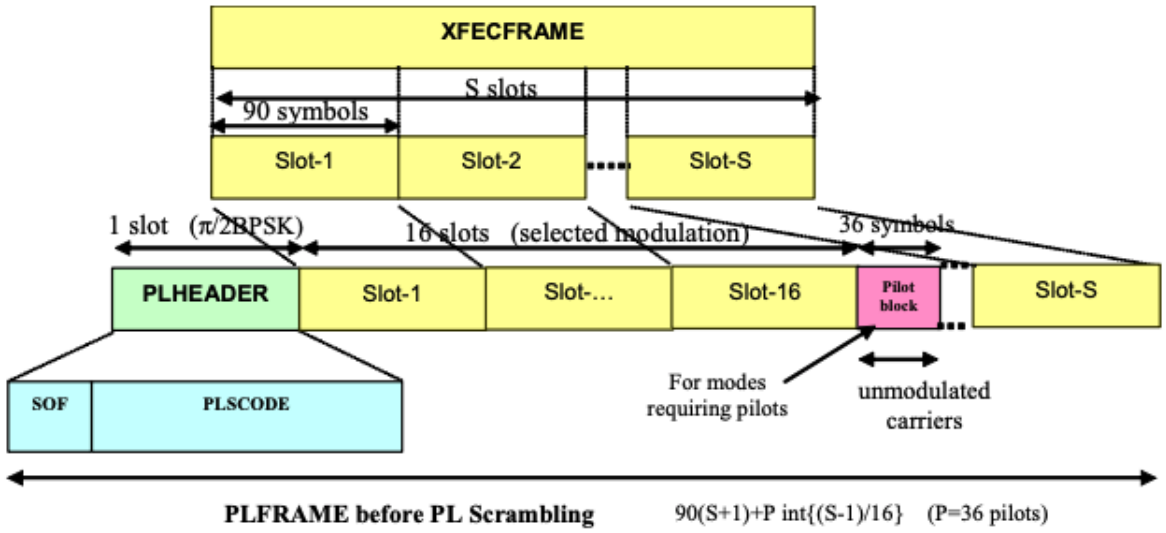

Figura 2.14: Modelo Trama PLFRAME [6]

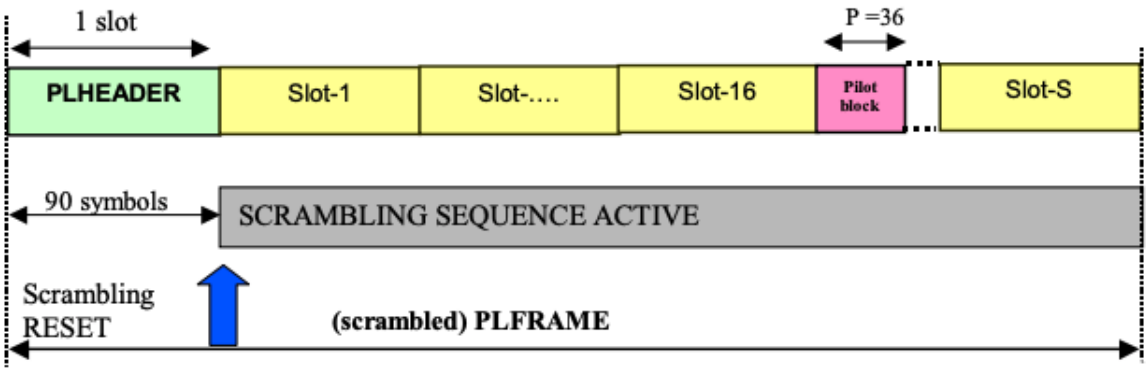

Figura 2.15: Esquema PLFRAME Mezclada [6]

considerar una degradación adicional de 0.2-0.3 [dB] [6].

2.2 Conclusión

Este capítulo está orientado a dar una base teórica del sistema DVB-S2 y la función de cada una de sus partes ya que es el sistema que se va a implementar en la Demo de Matlab que se especificará en el capítulo 4. Es fundamental conocer como y en que punto se produce la decodificación y demodulación de estas señales DVB-S2 de vídeo y cuales son los codificadores utilizados; en este caso el codificador LDPC. En resumen el sistema DVB-S2 realiza las siguientes operaciones [18]:

1. Generación de BBFRAME por una fuente aleatoria.

2. Codificación BCH, para todos los parámetros de codificación y FECFRAME normal.

3. Codificación LDPC, para todos los parámetros de codificación y FECFRAME normal.
4. Entrelazado.
5. Modulación.
6. Modelado de canales AWGN.
7. Demodulación de decisión suave.
8. Desintercalado.
9. Decodificación LDPC.
10. Decodificación BCH.
11. Eliminación del *buffer* de BBFRAME.

En el capítulo siguiente se introducirá el campo de las redes neuronales desde un punto de vista teórico para poder contextualizar al lector de cara a la Parte 4 y 5 donde se desarrollará la Demo y se presentarán los resultados.

Mode	Spectral efficiency	Ideal E_s/No (dB) for FECFRAME length = 64 800
QPSK 1/4	0,490243	-2,35
QPSK 1/3	0,656448	-1,24
QPSK 2/5	0,789412	-0,30
QPSK 1/2	0,988858	1,00
QPSK 3/5	1,188304	2,23
QPSK 2/3	1,322253	3,10
QPSK 3/4	1,487473	4,03
QPSK 4/5	1,587196	4,68
QPSK 5/6	1,654663	5,18
QPSK 8/9	1,766451	6,20
QPSK 9/10	1,788612	6,42
8PSK 3/5	1,779991	5,50
8PSK 2/3	1,980636	6,62
8PSK 3/4	2,228124	7,91
8PSK 5/6	2,478562	9,35
8PSK 8/9	2,646012	10,69
8PSK 9/10	2,679207	10,98
16APSK 2/3	2,637201	8,97
16APSK 3/4	2,966728	10,21
16APSK 4/5	3,165623	11,03
16APSK 5/6	3,300184	11,61
16APSK 8/9	3,523143	12,89
16APSK 9/10	3,567342	13,13
32APSK 3/4	3,703295	12,73
32APSK 4/5	3,951571	13,64
32APSK 5/6	4,119540	14,28
32APSK 8/9	4,397854	15,69
32APSK 9/10	4,453027	16,05
NOTE: Given the system spectral efficiency η_{tot} the ratio between the energy per information bit and single sided noise power spectral density $E_b/N_0 = E_s/N_0 - 10\log_{10}(\eta_{tot})$.		

Figura 2.16: Rendimiento Es/No sin casi error y PER= 1e-8 [6].

Capítulo 3

Redes Neuronales

3.1 Contexto Histórico

Si se tuviese que decidir quien fue el primer descubridor de las redes neuronales se tendría que retroceder en el tiempo hasta el año 1888 cuando Ramón y Cajal descubrió la neurona y el conjunto de neuronas interconectadas entre sí que dan lugar a las redes neuronales [14]. Se puede afirmar que Ramón y Cajal fue el descubridor de la red neuronal primigenia y que por tanto su origen está ligado al ámbito de la medicina.

Ya en la década de los años treinta los científicos empezaron a tener inquietudes acerca de la Inteligencia Artificial y sus posibles implantaciones en el sector industrial y con el fin de automatizar cadenas de producción surgieron las redes neuronales.

Fue en 1936 la primera reseña histórica conocida cuando Alan Turing inició sus investigaciones acerca del estudio del cerebro visto desde un punto tecnológico, el de la computación, pero no fue hasta 1943 cuando Warren McCulloch y Walter Pitts expusieron la primera red neuronal primitiva y por lo tanto muy simple mediante circuitos eléctricos en su teoría[19] acerca del "funcionamiento" de las neuronas. Modelo "McCulloch-Pitts".

Seis años más tarde, en 1949 sería Donald Hebb sería el primero en explicar el proceso de aprendizaje de una red neuronal con la regla Hebbiana de aprendizaje, el cual afirma que "el aprendizaje ocurría cuando ciertos cambios en una red neuronal eran activados" [17].

El siguiente hecho remarcable fue en 1950 cuando Karl Lashley, psicólogo conductista estadounidense, tras numerosos ensayos llego a la afirmación de que la información se encontraba distribuida en la corteza cerebral o cortex del cerebro humano [29].

Ya en la década de los 50, se dieron tres acontecimientos importantes. El primero fue en 1956 con la conferencia de Dartmouth, la cual fue organizada por Fue organizada por *John McCarthy*

(Dartmouth College, Nuevo Hampshire), Marvin L. Minsky (Harvard University), Nathaniel Rochester (I.B.M. Corporation) y Claude E. Shannon. Fue organizada en Hanover y se invitaron a 10 investigadores prestigiosos con el fin de hacer grandes avances acerca de la inteligencia artificial durante 2 meses [28]. Esta conferencia se utiliza para fechar el nacimiento de la inteligencia artificial; después entre los años 1957-1958, Frank Rosenblatt inicia los estudios de lo que hoy se conoce como Perceptrón [17].

Ya en la década de los 60 cabe estacar el artículo en 1961 de Karl Steinbeck y el libro "Perceptrones" de Marvin Minsky y Seymour Papert, ambos demuestran que el perceptrón es demasiado simple/débil demostrando que no es capaz de resolver/aprender funcionas no lineales lo cual suponía una limitación dado que muchos de los problemas matemáticos más relevantes involucran funciones no lineales.

En la década de los 70, *Paul Werbos* introdujo el *backpropagation* como mecanismo de aprendizaje y en 1977 Stephen Grossberg formuló la Teoría de Resonancia Adaptada (TRA).

Tras varios años donde se estancaron las investigaciones sobre este ámbito, en 1982 Kohonen desarrolla los mapas auto-organizativos. No es hasta el 1986 con el aporte de David Rumelhart cuando se relanzan de nuevo las investigaciones con el redescubrimiento del *backpropagation* [21]. Hoy en día es un ámbito que se sigue investigando dado que es un campo con mucho potencial y futuro ante la nueva generación 5G y el IoT. Es de destacar el desarrollo de una red llamada BAM (Bidirectional Associated Memory), así como los logros obtenidos por IBM con su proyecto *"Deep blue"*, Apple con el lanzamiento de Siri o automóviles autónomos como Waymo de Google [3] [25].

3.2 Fundamentos Principales de una Red Neuronal

Como se ha comentado anteriormente, las Redes Neuronales Artificiales (RNA) son un concepto basado en el cerebro humano, el cual tiene cientos de millones de neuronas interconectadas entre sí de tal forma que son capaces de transmitir pulsos nerviosos de unas a otras activándose de manera consecutiva.

Las señales recibidas por la neurona son procesadas y elaboran una respuesta en consecuencia que se ve reflejada en el nivel de activación de la propia neurona. Esta es una analogía exacta del funcionamiento de una red neuronal, y fue en el año 2008 cuando *Haykin* elaboró un modelo de RNA: en este modelo se evalúa la unidad de una Red Neuronal. En la Figura 3.1 y como se puede observar se compone de una serie de entradas xj de donde procede la señal y son ponderadas en las wj donde se "pesa" la señal que es mandada a unión sumadora. Cada neurona posee un valor umbral propio, este valor hará que la neurona se active o que por el contrario no lo haga. "La

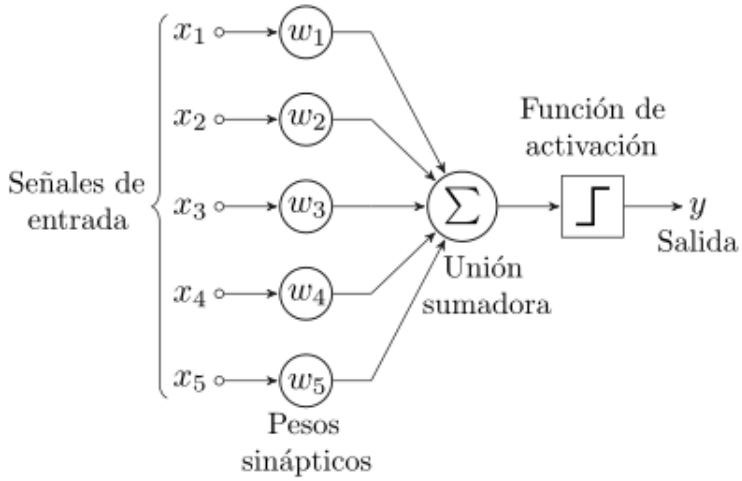

Figura 3.1: Red Neuronal Básica: El Perceptrón [30].

suma ponderada de las entradas menos el valor umbral componen la "activación" de la neurona" [16]. Después se encuentra la Función de activación la cual produce la señal de salida y evita que la salida tome valores no deseados. Para poder realizar esta estimación, la función de activación o también llamada función de transferencia habitualmente tuliza las siguientes funciones:

- Sigmoidea:
$$\sigma(x) = \frac{1}{1 - e^{-x}} \tag{3.1}$$
- Lineal:
$$\sigma(x) = x \tag{3.2}$$
- Escalón:
$$\sigma(x) = \begin{cases} 1 & \text{si x>=0} \\ 0 & \text{si x<0} \end{cases} \tag{3.3}$$
- Tangente hiperbólica:
$$\sigma(x) = \tanh(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{3.4}$$

Por lo tanto podríamos sintetizar la salida de una neurona mediante la fórmula:

$$y_j = \sigma(\sum_{i=1}^{N} w_{ij} x_i \pm \theta_j) \tag{3.5}$$

Por último es imprescindible tener una idea global de la composición de un sistema neuronal, esta recogido en un modelo realizado por Rumelhart y McClelland (1986) la cual queda reflejada en la Figura 3.2 y donde se explica el "despiece"de un sistema neuronal dividéndolo en la unidad mínima anterior que lo forma. De este modo una capa la forma una agrupación de neuronas y a su vez una red la forman una agrupación de capas, formando finalmente un sistema neuronal reuniendo redes.

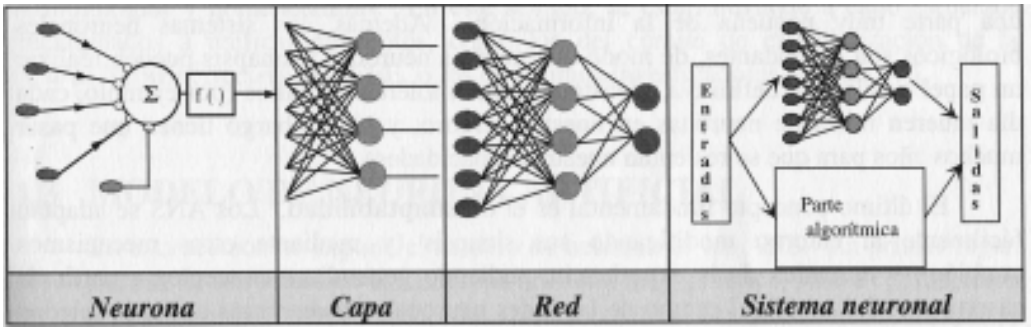

Figura 3.2: Visión Global Sistema Neuronal [14]

3.2.1 Beneficios de las redes neuronales

Las RNA han tenido un gran impacto en la sociedad moderna por su capacidad de resolver problemas de gran dificultad, imprecisos o costosos. Son capaces de resolver estos problemas complejos, generalmente no lineales, extrayendo patrones o creando tendencias de éxito con un alto porcentaje de éxito.

Algunas de las ventajas de las RNA son:

1. Aprendizaje Adaptativo: las RNA aprenden en función de los datos con las que se las entrenen.

2. Auto-organización: como su nombre indica, una RNA es capaz de organizar la información recibida de manera autónoma.

3. Operación a tiempo real: es decir una RNA puede estar procesando datos a su entrada procedentes de procesos distintos, ejecutándolos en paralelo.

4. Tolerancia a fallos vía codificación de la información redundante: "la destrucción parcial de una red lleva a la correspondiente degradación del rendimiento. Sin embargo, algunas capacidades de la red pueden ser retenidas a pesar de provocar importantes daños a la red"[16].

Hoy en día hay una serie de características que hacen que las RNA tengan una demanda exponencial y que hacen que sea inimaginable la vida sin ellas:

- Potencia: son instrumentos que se ha comprobado que son lo suficientemente potentes como para resolver funciones complejas así como funciones no lineales. Además, las redes neuronales también controlan la dimensionalidad, "que frustra los intentos de modelizar funciones no lineales con gran número de variables"[16].

- Facilidad de uso: antes se ha mencionado que las RNA tienen un aprendizaje adaptativo y aprenden a través del entrenamiento. El entrenador de estas redes neuronales debe tener

los conocimientos para entrenar correctamente a la red y saber seleccionar los datos de entrada, en definitiva la comparativa entre los conocimientos mínimos a adquirir y la resolución de las complejas funciones, en muchos casos no lineales hace que resulte sencillo resolver estas aparatosas y costosas funciones.

- Facilidad de implementación hardware: la operación en tiempo real y por lo tanto la característica de operar en paralelo hace que se reduzca considerablemente el tiempo de espera a la respuesta de salida, es por ello que se implementan con tecnología VLSI, la cual proporciona un método para capturar comportamientos muy complejos de manera jerárquica [15].

3.3 Estudio del entrenamiento de una red neuronal

A la hora de entrenar de las Redes Neuronales, existen tres formas básicas de aprender: Aprendizaje Supervisado, No Supervisado y Aprendizaje Reforzado. Este último va a caballo entre los dos primeros, por lo que fundamentalmente se especificarán los dos primeros tipos de aprendizaje.

3.3.1 Supervisado

Este tipo de aprendizaje está basado en un circuito que se retroalimenta como podemos ver en la Figura 3.3 y en este caso, es vital saber la salida deseada, como se mostrará a continuación. Las entradas llegan a la RNA y se elabora la salida de la RNA, está salida es comparada con la deseada y se determina si el resultado es correcto o incorrecto y se restablecen los pesos (los cuales están inicializados a cero) conforme a los resultados obtenidos de manera iterativa hasta que se llega a la salida de la RNA deseada. "Los pesos se van modificando de manera proporcional al error que se produce entre la salida de la red y la salida esperada"[23].

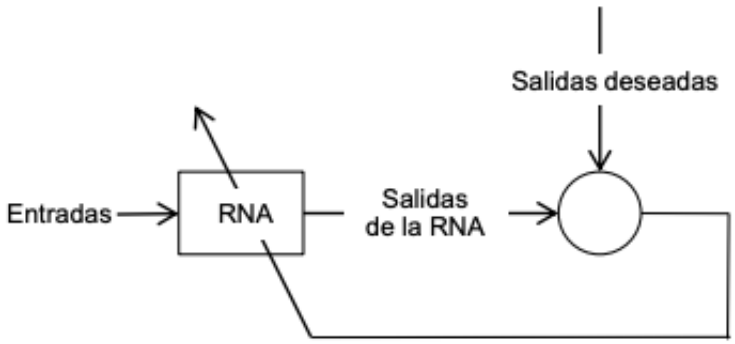

Figura 3.3: Aprendizaje Supervisado [23]

3.3.2 No Supervisado

También llamadas auto-organizativas o redes competitivas cuyo funcionamiento está fundado en técnicas de *clustering*, la cual çonstituyen una técnica exploratoria, útil para encontrar conglomerados de casos que se puedan unir de acuerdo con un grupo de variables"[4] donde se asocian muestras de características similares en un mismo *cluster*. Es decir, "La idea inherente de todas estas redes es que la capa oculta de neuronas debe ser capaz de extraer las características estadísticas del conjunto de datos de entrada"[16]. Generalmente dentro de la capa oculta de estas redes se producen rivalidades entre las neuronas para determinar cual es la mas apropiada que es la que finalmente se actualizará [21].

3.4 Tipos de Redes Neuronales

3.4.1 Perceptrón Simple

En año 1962 Rosenblatt hace públicos sus estudios acerca de lo que se conoce como la primera red neuronal, el perceptrón. Se trata de la red neuronal más básica y es usada generalmente como clasificador de dos clases linealmente separables como, por ejemplo, vectores con valores binarios.

El perceptrón esta formado por una única neurona como se ve en la Figura 3.1. y tiene por ecuación de salida:

$$S_j = \sum w_{ij} a_i \tag{3.6}$$

Donde S_j es la señal de salida del perceptrón, w_j serán los pesos sinápticos y a_j cada entrada de datos al perceptrón. Fijándose en la Fórmula 3.6, cuando el perceptrón evalúa la señal de entrada y transmite un 1 o un 0 en función del valor umbral. La principal limitación del perceptrón simple es no poder resolver problemas cuya entrada no sea linealmente separable, lo cual lo limita considerablemente y lo hace poco útil[23] [16] [14].

- **Algoritmo de Aprendizaje**: tomando como referencia la Figura 3.1, a cada peso sináptico, w_j, se le va a dar un peso inicialmente aleatorio, que se irá cambiando automáticamente, debido al proceso iterativo, si la salida no corresponde con la salida esperada. La regla de aprendizaje del Perceptrón Simple viene dada por la Fórmula 3.7:

$$w_j(k+1) = w_j(k) + \eta(k) * [z(k) - y(k)] * x_j(k) \tag{3.7}$$

Donde "la variación del peso w_j es proporcional al producto del error z(k)y(k) por la componente j-ésima del patrón de entrada que hemos introducido en la iteración k, es decir, x_j(k). La constante de proporcionalidad η(k) es un parámetro positivo que recibe el nombre de tasa de

aprendizaje, ya que cuanto mayor es, más se modifica el peso sináptico y viceversa" [20]. Pasos en el algoritmo de aprendizaje:

1. Inicialización: se da un valor aleatorio a los pesos sinápticos.
2. (k-ésima iteración): calcular

$$y(k) = sgn * (\sum_{j=1}^{n+1} w_j x_j(k)) \tag{3.8}$$

3. Corrección de los pesos sinápticos: si z(k)≠y(k) se deberá modificar los pesos según la Fórmula 3.7.
4. Parada: si los pesos siguen igual y no han variado, la red se ha estabilizado. Si por lo contrario siguen variando, volver al paso 1.

3.4.2 Perceptrón Multicapa

Se trata de una RNA unidireccional, compuesta por mínimo 3 capas, formadas a su vez por perceptrones simples.

Como se puede observar en la Figura 3.4, un Perceptrón Multicapa está formado por una capa de entrada, N capas ocultas, y una capa de salida.

La capa de entrada recibe las señales y las introduce en la red neuronal, es una capa que no tiene ni pesos ni umbrales y no se realizan operaciones de ningún tipo, por lo que no es considerada una capa de neuronas. Esta capa simplemente se encarga de procesar datos a la entrada y distribuirlos por toda la red neuronal [23].

El número de capas ocultas dentro de una RNA es importante que esté optimizado para evitar pérdida de recursos.

Tras numerosos experimentos y pruebas se ha visto que el rendimiento de un Perceptrón multicapa, que tiene una sola capa oculta mejora significativamente si se añade una segunda capa.

Por otro lado, es importante apuntar que cada neurona esta interconectada con cada neurona de la n+1 columna de la siguiente capa, aunque se pueden dar excepciones donde una neurona solo se interconecta con un subconjunto de neuronas de la siguiente capa [20].

Finalmente, la capa de salida transmite el resultado de la función de activación de cada neurona en esta capa de modo que se dispondrá de tantas salidas como variables de salida tenga el problema a considerar.

El algoritmo de entrenamiento de esta red será el de el perceptrón simple llevado a gran escala al poseer más de una capa.

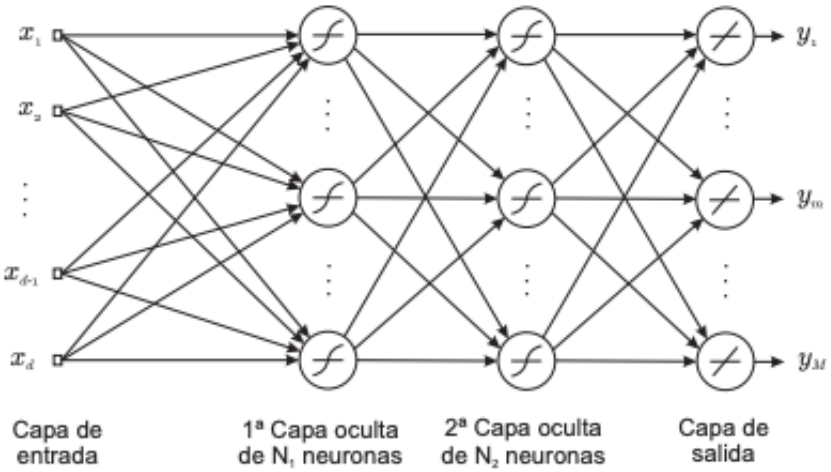

Figura 3.4: Modelo Perceptrón Multicapa [16].

3.4.3 La Red de funciones de Base Radial

Las redes de funciones de Base Radial nacen de las investigaciones acerca de la interpolación en espacios multidimensionales (método de interpolación exacta), de esta forma se consigue aproximar todos los puntos de un conjunto de datos de manera exacta. Esto es gracias a una serie de funciones base centradas en ciertos puntos que, al ser combinadas linealmente, hacen el papel de aproximadores locales [16].

Este método requiere que haya un nodo por cada muestra de entrada, lo que hace que su coste se encarezca debido al aumento de gasto de recursos. Además es un método muy susceptible al ruido, es decir, al tratarse de un método exacto de aproximación a los puntos del conjunto de entrenamiento, el ruido hará que la exactitud no sea del 100 %. Tras una serie de modificaciones se obtiene el modelo de funciones de base radial (RBFN) donde el número de funciones base está determinado por la complejidad del mapeo a representar.

1. El número de funciones base, N, no necesita ser igual al número de muestras del conjunto de datos.

2. Los centros de las funciones base se modifican durante el proceso de entrenamiento.

3. Cada función base tiene su propia anchura, que también es modificada en el entrenamiento.

4. Los umbrales se incluyen en la suma lineal realizada en los nodos de salida.

La RBFN está conformada por tres capas: capa de entrada, capa oculta y capa de salida. Es en la capa oculta donde reside la gran diferencia entre las RBFN y los Perceptrones Multicapa, dado que "se trata de una capa no lineal que aplica una transformación no lineal a los datos

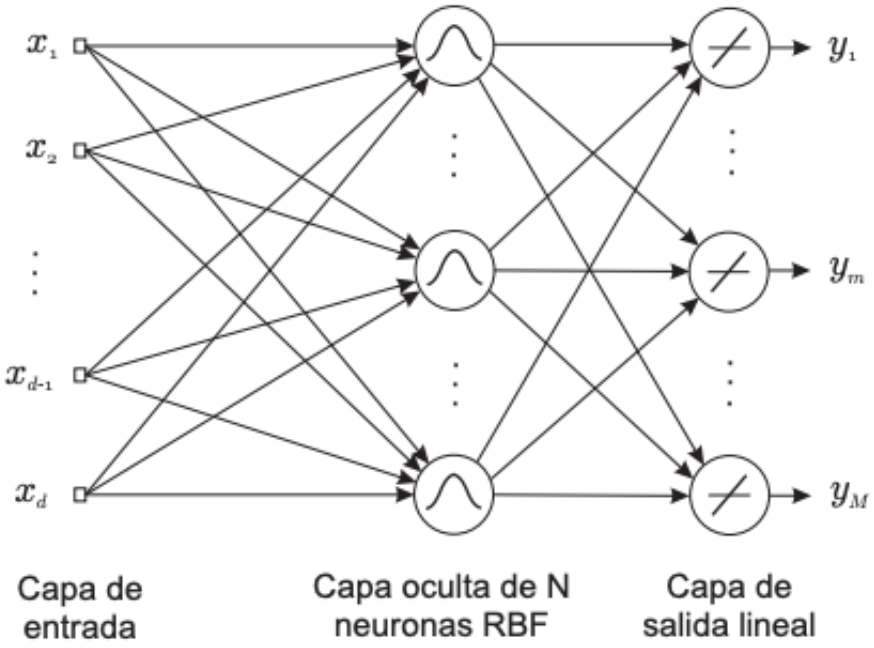

Figura 3.5: Red de Funciones de Base Radial [16]

precedentes de las entradas a un espacio habitualmente de mayor dimensionalidad. La capa de salida es lineal y proporciona los valores estimados de las variables objetivo"[16].

Algunas de las diferencias entre Red de Base Radial y los Perceptrones Multicapa son [27]:

- Distintos mecanismos de Aprendizaje (RBNF se realiza en dos etapas).
- En las RBFN las conexiones entre la capa de entrada y la capa oculta son conexiones directas sin pesos.
- En los Perceptrones Multicapa el número de capas ocultas no esta delimitado, sin embargo en las RBFN solo hay una capa oculta.
- Los Perceptrones Multicapa calculan aproximaciones "globales"del mapeo no lineal entre entradas y salidas, sin embargo los RBNF calculan aproximaciones "locales"del mapeo entre entradas y salidas.

Algoritmo de Aprendizaje: Para poder definir el mecanismo de aprendizaje primero hay que definir la función que desarrolla para obtener la k-esima variable de salida:

$$y_k(x) = \sum_{N}^{j=1} w_{kjx} + b_k \tag{3.9}$$

De la función podemos identificar a ϕ_j como la función base del j-ésimo nodo; w_{kj} es el peso asociado a la conexión entre la neurona j de la capa oculta y la neurona k de la capa de salida

y b_k es el umbral de la neurona k de la capa de salida [16]. A su vez la fórmula de la base (en este caso Gaussiana) utilizada es:

$$\phi_j(x) = e^{-\frac{\|x-\mu_j^2\|}{2\sigma_j^2}} \tag{3.10}$$

Donde μ_j es le vector que representa el centro de la función base ϕ_j(x) [16]. Una vez definidas estas dos funciones se puede decir que el entrenamiento consta de dos fases. En la primera se utiliza el conjunto de datos de entrada, de manera no supervisada (sin utilizar las salidas), para determinar los parámetros (μ_j y σ_j) de las funciones base. En la segunda fase estos parámetros calculados se fijan y es ahora cuando se calculan los pesos de la capa por medio de aprendizaje supervisado [16].

3.4.4 Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial o *Support Vector Machine* (SVM) su funcionamiento está fundamentado en funciones Kernel y en la teoría de aprendizaje estadístico y en la dimensión Vapnic-Chervonenkis [1].

La base de una SVM es construir hiper-planos como superficie de decisión que separa con máximo margen las muestras de una y otra clase, se puede ver representado en la Figura 3.6.

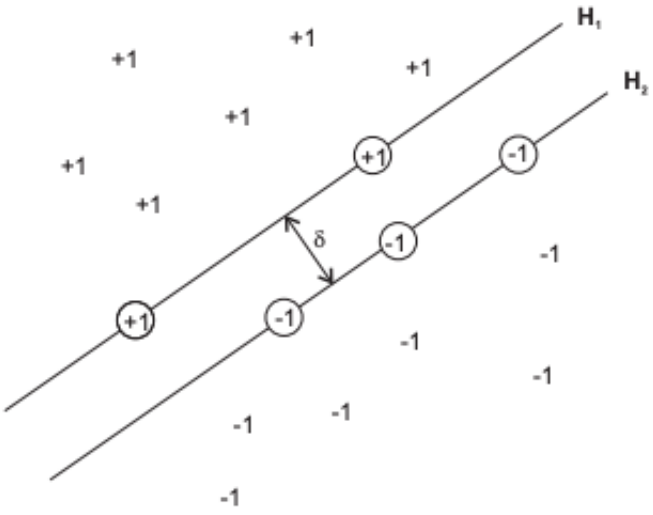

Figura 3.6: Hiperplano para un problema con dos clases [16]

Como se puede observar en la Figura 3.7 los mapas de soporte vectorial están formados por una capa de entrada de tamaño N, seguida de una capa oculta con el mismo número de entradas. Cada señal entrante de las neuronas de la capa de entrada se interconecta con cada neurona de la capa oculta.

Finalmente las salidas de las capas ocultas convergen en un punto, concretamente en la Neurona

de Salida, junto con el umbral 'b' propio de la Neurona de Salida, la cual elabora la señal de salida y.

La principal función de las SVM es la resolución de problemas de clasificación mediante la construcción de hiper-planos en un espacio multidimensional."Para construir un hiper-plano óptimo, se emplea un algoritmo iterativo de entrenamiento, el cual minimiza la función de error"[16].

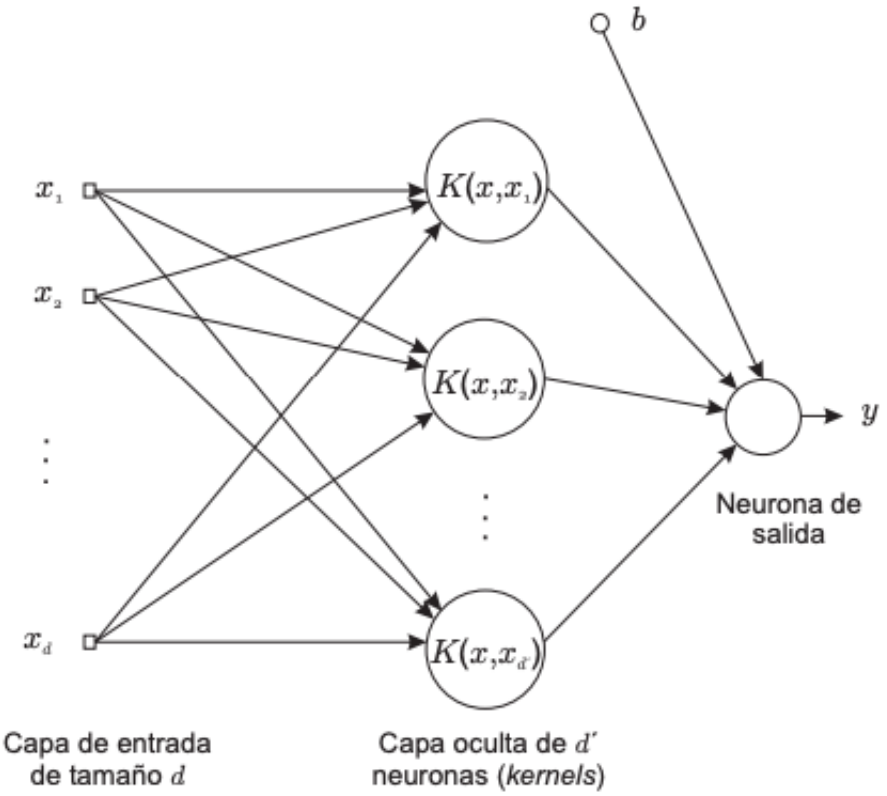

Figura 3.7: Estructura de una Máquina de Soporte Vectorial [16].

"Las SVMs se basan en dos operaciones fundamentales. La primera es un mapeo de los vectores de entrada a un "espacio de características" de mayor dimensionalidad, que sólo se emplea de manera ocasional y está oculto tanto para las entradas como para la salida. La segunda consiste en la construcción del hiper-plano óptimo de separación de las características encontradas en el paso anterior."[16] De esta forma, mediante las SVM se es capaz de separar las clases de datos que a priori eran inseparables linealmente en el espacio de entrada, gracias a un nuevo espacio llamada espacio de características [27].

3.4.5 Los Mapas Auto-Organizativos

El creador de los Mapas Auto-Organizativos o *Self-Organizing Maps* (SOM) fue Kohonen, los cuales fueron desarrollados en 1982 como se ha comentado en el punto 3.1. SOM es un mapeo

artificial que aprende de la auto-organización. Este tipo de red neuronal posee una capa donde las neuronas que la componen compiten entre sí para obtener de manera más exacta el patrón de entrada buscado como queda representado en la Figura 3.8.

Cuando el entrenamiento de la red haya concluido habrá subconjuntos de neuronas que hayan encontrado características similares las cuales quedarán recogidas en las neuronas correspondientes con la singularidad de que de manera sistemática siempre activarán las mismas neuronas de salida. Es a este proceso, que realizan las neuronas, lo que se conoce como mapeo de los datos que almacenan.

"Cada una de las neuronas artificiales posee un vector de pesos, el cual será, una vez finalizado el entrenamiento, el vector prototipo que representará a los patrones de entrada que mapee22 [11]. Estas neuronas que conforman esta capa competitiva siguen un entrenamiento no supervisado como ya se ha mencionado en el punto 3.3.2.

De esta forma la red es capaz de conocer por sí misma las características de los datos de entrada.

- **Entrenamiento de la red**: Para entrenar a la red se introducen patrones de entrada los cuales se cotejan con los pesos de los vectores de las neuronas que conforman la red. Estos pesos son atribuidos previamente al entrenamiento de la red.

A partir de los resultados obtenidos se dictamina una neurona ganadora, que será la que tenga los resultados mas similares a los esperados de tal forma lo cual provoca una reacción en cadena haciendo que las neuronas vecinas hagan un reajuste de sus pesos pareciéndose a la neurona ganadora.

De manera iterativa se repite este proceso hasta que se logre que el subjconjunto entero tenga patrones de entrada similares [11].

3.4.6 Red Backpropagation

3.4.7 Introducción

En la época de los años setenta se produjo un estancamiento en cuanto a las investigaciones de las redes neuronales y , en concreto, a los algoritmos de entrenamiento de las mismas. El descubrimiento de la red Backpropagation supuso un antes y un después en el ámbito de las RNA, que hizo avivar el interés por este campo y abrir nuevas puertas a la investigación.

Arquitectura de la Red Backpropagation

Individualmente, las neuronas que conforman la Estructura de la RBP son neuronas simples que reciben una serie de entradas y elaboran una salida. Se caracterizan por tener un parámetro θ

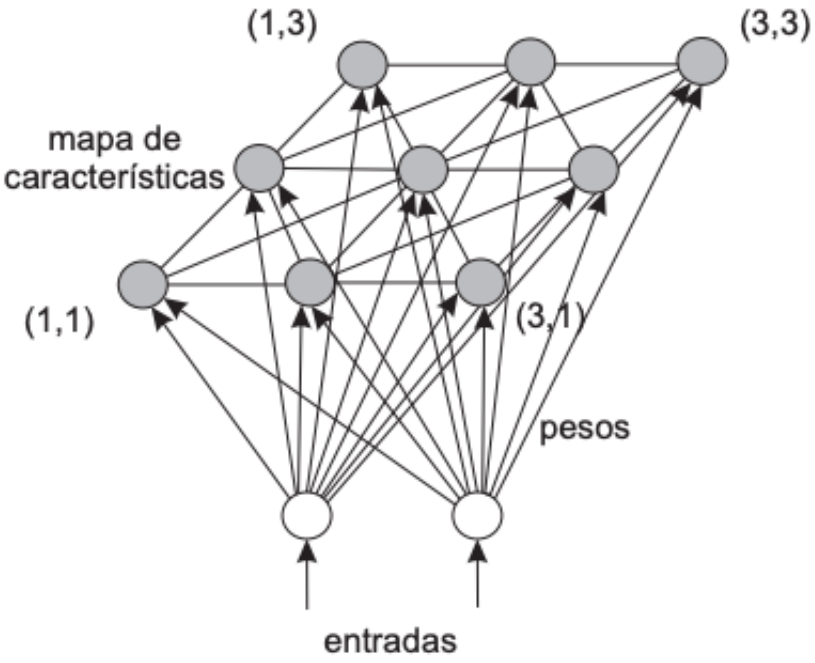

Figura 3.8: Estructura de un mapa auto-organizativo con 2 entradas y un mapa de salida 3x3 [16]

que será imprescindible en el ajuste de pesos posterior. Estos pesos pueden variar a lo largo del proceso de aprendizaje. Lo normal es que las RBP presenten más de 3 capas procesadoras. En el ejemplo de la Figura 3.10 se puede diferenciar una RBP de 3 capas. La capa de entrada, la cual no realiza cálculos numéricos ni maneja pesos en las neuronas que lo componen. La capa oculta, la cual salta a la vista que todas las neuronas que la conforman están interconectadas como la capa superior como con la inferior. Finalmente la capa superior elabora la salida en consecuencia [21].

Algoritmo de Entrenamiento

El algoritmo de de entrenamiento de una RBP sigue un aprendizaje supervisado y consta de dos partes: una fase de propagación hacia adelante y otra fase hacia atrás.

- Inicialización: en el caso de que no exista información preexistente, los valores de los pesos y umbrales serán inicializados con valores aleatorios pequeños.
- Presentación de las muestras de entrenamiento: se introducen a la entrada dela RBP muestras de entrenamiento correspondientes a un epoch, se conoce como epoch a una presentación completa del conjunto de entrenamiento durante el proceso de aprendizaje. Posteriormente se procederá a la fase *forward* o de propagación hacia delante y la fase *backward* o propagación hacia atrás, como se detella a continuación.
- Propagación hacia Adelante o Fase *Forward*: Una vez que se ha aplicado un patrón a la

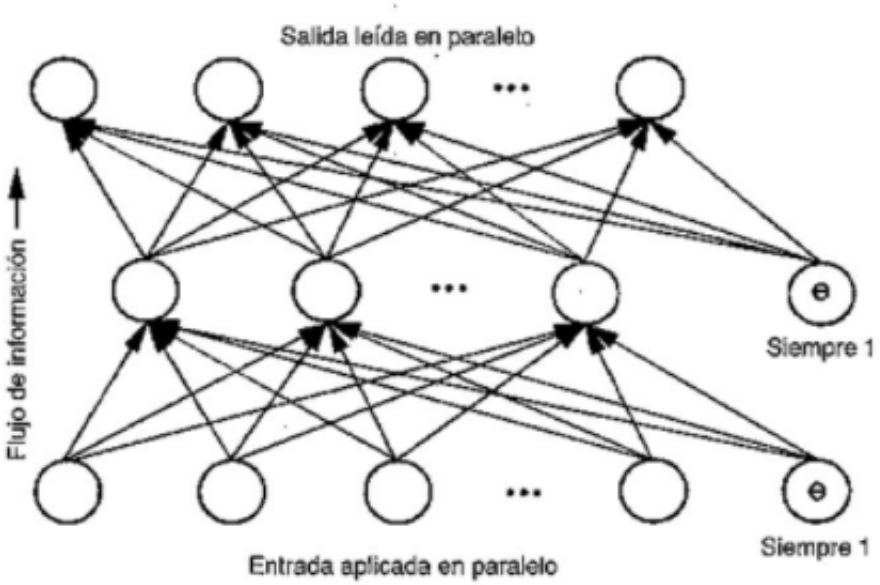

Figura 3.9: Arquitectura Red Backpropagation

entrada de la red como estímulo, éste se propaga, activando a su paso las capas superiores en cada caso, desde la primera capa hasta las siguientes, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las variables de [23].

- Propagación hacia atrás o Fase *Backaward*: esta fase se inicia cuando la fase forward haya finalizado. En este punto se realiza el proceso contrario. Las señales de error se propagan hacia atrás, siguiendo el camino que habían tomado previamente par allegar a la capa de salida, de este modo cada neurona de cada capa recibe una parte del error total basándose en la contribución que haya tenido esa neurona a la salida original. "Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita aprender correctamente todos los patrones de entrenamiento. La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas aprenden a reconocer distintas características del espacio total de entrada" [23].

  Los ajustes de estos pesos se pueden dividir en dos clases:

  - Ajuste de Pesos de la Capa de Salida: como ya se ha mencionado anteriormente en este punto la RBP sigue un proceso de aprendizaje supervisado por lo que se conoce el valor deseado de salida y basta con comparar y examinar ambos resultados.
  - Ajuste de Pesos de las Capas Ocultas: Los pesos serán ajustados según la ecuación postulada por *Rumelhart y McClelland*:

$$\delta w_{ij} = \mu \delta_j a_i \tag{3.11}$$

Donde w_{ij} será la variación de los pesos de las neuronas; μ corresponderá al coeficiente de aprendizaje, el cual suele tomar un valor entre 0.25 y 0.75; δ_j representará el error parcial asociado a cada neurona; y por último a_i que representa el valor de entrada correspondiente.

Convergencia: al entrenar una RBP es habitual ver el uso del parámetro RMS (*Root Mean Square*) para ser conscientes del grado de error de la red o, lo que es lo mismo, conocer el grado de acierto. "Es necesario definir un parámetro de corte o un valor umbral del valor RMS del error de la red que permita decir que la red se aproxima a la salida deseada y considerar que la respuesta es correcta" [21].

Una vez visto el funcionamiento de las redes neuronales con el nivel de detalle que necesario para este TFG se procede en la siguiente parte a presentar la Demo de Matlab, herramienta fundamental del trabajo bajo el que se sustenta la parte experimental, la cual está dividida en dos partes y que se. verán a continuación.

Capítulo 4

Estudio Demo Matlab

4.1 Base Teórica de la Demo

El objetivo final de esta Demo, y en lo que se basa finalmente este TFG, la cual se detalla en el Anexo, es ver como de precisa es nuestra red neuronal (LLRNet) a la hora de demodular la señal de entrada frente a la alternativa de calcular las LLR de forma exacta. Es conveniente poder utilizar estas redes neuronales debido a que calcular la LLR de manera exacta (ver Figura 4.2) supondría un gasto de recursos enormes en la implementación real y conduciría a redondeos innecesarios, pero sobre podría haber problemas de estabilidad numérica por el uso de exponenciales.

Es importante conocer en que punto de la cadena de recepción está situada los cálculos realizados en esta Demo. Para saber situar en que punto se encuentra las red neuronal puede servir de ayuda fijarse en la Figura 2.4.

En resumen el objetivo de usar RNA aquí es evitar carga de implementación.

El principio matemático que define LLR es el logaritmo de la relación entre la probabilidad de que un bit sea 1 o que por el contrario sea 0 y se calculará entre la salida del canal AWGN y el decodificador LDPC.

$$l_i \triangleq \log\left(\frac{P_r(c_i = 0|\widehat{s})}{P_r(c_i = 1|\widehat{s})}\right), i = 1, \ldots, k$$

Figura 4.1: Principio Matemático LLR [18]

De esta forma, se van a comparar los resultados obtenidos de tres formas diferentes:

- LLR exacta: como antes se ha mencionado, esta es la manera exacta de, al demodular un símbolo, conocer los bits que lo conforman. Al ver la Figura 4.2 y 4.3 se puede observar

que es una fórmula bastante compleja formada por logaritmos y sumatorios de exponenciales, por lo que la implementación real de la LLR Exacta en un dispositivo consumiría demasiados recursos.

Como se puede observar se va a en el numerador se va a calcular el logaritmo de la probabilidad de que un determinado bit de la palabra codificado sea 0 bajo la hipótesis frente a la hipótesis de que sea un 1. De tal forma que al recibir un símbolo hay que estimar los bits de ese símbolo, para ello se calcula una probabilidad basada en la distancia a cada símbolo teórico. Además tanto en el numerador como en el denominador la distancia previamente mencionada entre el símbolo teórico y el símbolo recibido está dividido por la potencia de ruido.

La potencia de ruido es un factor con una gran relevancia. En un LLR es indispensable estimar la potencia de ruido sigma porque cuanto mayor sea el ruido en el canal, los resultados serán más imprecisos y menos fiables.

$$l_i \triangleq \log\left(\frac{\sum_{s\in c_i^0}\exp\left(-\frac{\|\hat{s}-s\|_2^2}{\sigma^2}\right)}{\sum_{s\in c_i^1}\exp\left(-\frac{\|\hat{s}-s\|_2^2}{\sigma^2}\right)}\right)$$

Figura 4.2: Cálculo de LLR Exacta [18]

- LLR Aproximada: definida en la Figura 4.4, funciona bien cuando el ruido es pequeño. Al realizarse esa aproximación con la LLR Aproximada si el resultado final es positivo, la probabilidad de que el bit sea 1 es mayor que la de ser 0 y si por el contrario es negativa será más probable que ese bit sea un 0.

$$\log\left(\sum_j \exp\left(-x_j^2\right)\right) \approx \max_j\left(-x_j^2\right)$$

Figura 4.3: Cálculo de LLR Aproximada [18]

$$l_i \approx \frac{1}{\sigma^2}\left(\min_{s\in C_i^1}\|\hat{s}-s\|_2^2 - \min_{s\in C_i^0}\|\hat{s}-s\|_2^2\right)$$

Figura 4.4: Cálculo de LLR Aproximada

- LLR Net: método de cálculo LLR basado en redes neuronales y que se tratará más adelante.

4.2 Parte 1

En la Demo primeramente se calcula a modo de ejemplo los valores de LLR Aproximada y de LLR Exacta para una 16QAM. Se evalúa como varían los valores de LLR Exacta y Aproximada: si se modifica el valor de la SNR, a priori si se reduce su valor, el valor de la LLR Aproximada será más parecido al de la LLR Exacta

Esta primera parte de la demo tiene varias subpartes. La primera es donde se van a introducir los datos iniciales que se necesitarán para posteriores operaciones. A continuación se procede a generar una constelación de N símbolos (valor configurable a la entrada) para ello el programa pasa por parámetro el orden de modulación M y una variable (sym0rder) que va a guardar el resultado de la generación de los símbolos. Mediante estos datos se genera una constelación de referencia.

Después hay que generar un ruido para emular una señal entrante al demodulador real para ello se requerirá del valor de la SNR para hallar tanto la varianza como sigma. Con esos datos de entrada más el número de símboloss se completa los parámetros de la función para generar el ruido en el canal correspondiente.

Para acabar se realiza la demodulación con la función *qamdemod* a la cual se tendrá que pasar por parámetros la potencia de ruido, el orden de modulación, la varianza de ruido y el sym0rder. De esta forma se obtiene los valores de la LLR Exacta y LLR Aproximada los cuales se compararán.

En esta parte se va a buscar la comparación de las gráficas obtenidas variando:

- Variación del orden de Modulación (M).
- Variación de la profundidad de la capa oculta de la RNA.
- Variación del número de símboloss.

Cabe destacar que se ha modificado el código, aumentando el número de intentos de la MSE, aumentándolo hasta llegar a 5 intentos.

En la Figura 4.5 se ilustra un diagrama de la Parte 1 de la Demo.

Se va a programar inicialmente una red neuronal con una capa de entrada con dos entradas: en la primera de ella colocaremos la parte real del símbolo recibido, la salida por el contrario la conformará un vector de dimensión kxN donde k es el número de bits por símbolo y N es el número de símbolos, así tenemos tantas LLR como bits transmitidos. En esta parte se va a variar

En cuanto a la capa oculta inicialmente estará configurada con una profundidad de ocho. A continuación teniendo como referencia la Figura 4.6 se muestra la estructura de la LLRNet, la

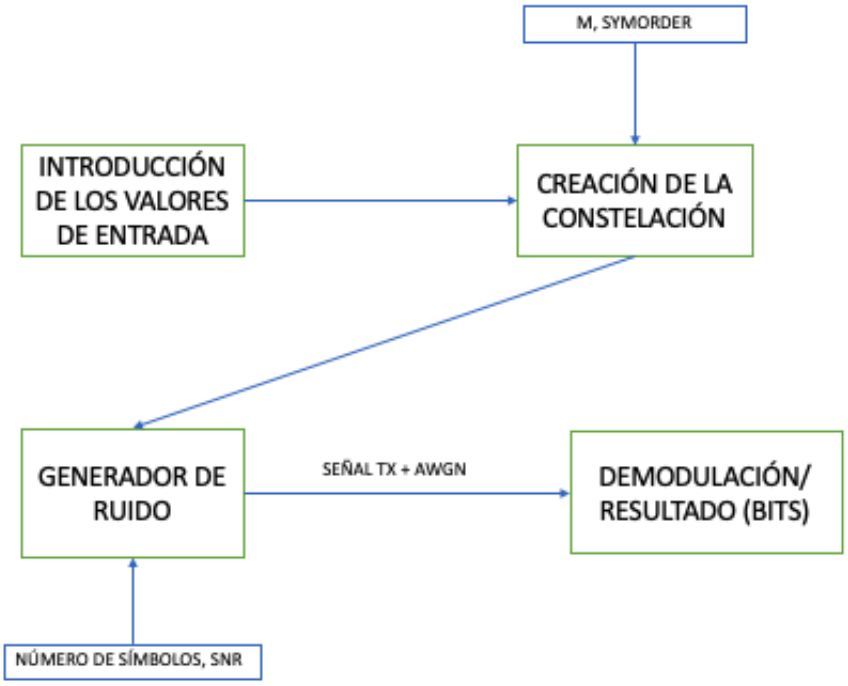

Figura 4.5: Diagrama del flujo de funcionamiento de la Parte 1.

cual corresponde a un Perceptrón Multicapa visto anteriormente en el Capítulo 3; se procede al entrenamiento, supervisado en este caso (fijarse como en la Figura 4.7 se produce retroalimentación, característica previamente mencionado en este tipo de entrenamiento), de la red neuronal donde el programa se encargará de entrenarla por medio de la función *llrnetNeuralNetwork* para ello la Demo está configurada para calcular el valor del error cuadrático medio (MSE). Se ha modificado el programa para que haga cinco pruebas en vez de tres (como viene por defecto) con el motivo de poder descartar pruebas con resultados no razonables. Se repetirá el proceso para cada valor de SNR. En la Figura 4.8 se representa a modo de aclaración un diagrama de bloques que explica el funcionamiento de esta parte de la Demo.

4.3 Parte 2

El sistema DVB-S2 utiliza un demodulador suave para generar entradas para el codificador LDPC. Por ello en esta parte vamos a calcular la *Packet Error Rate* (PER) para una modulación 16APSK y código LDPC 2/3, los cuales son los valores por defecto.

Se probará a variar el número de número de símbolos y profundidad de la capa oculta para comprobar el rendimiento en cada uno de los casos, sus pros y sus contras [18].

En la Parte 5, se expondrán los resultados de las simulaciones para observar como influyen los distintos parámetros de entrada en el rendimiento del sistema.

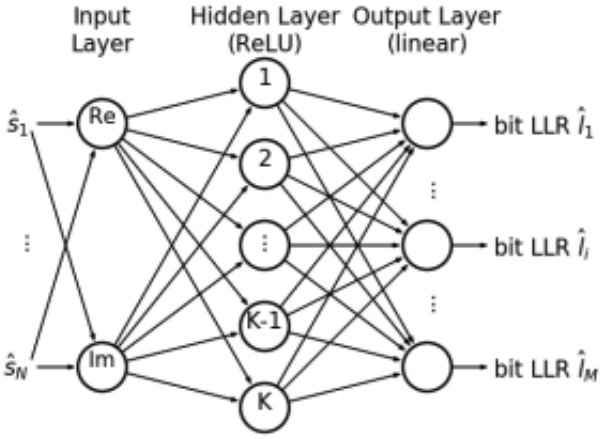

Figura 4.6: Estructura LLRNet [26]

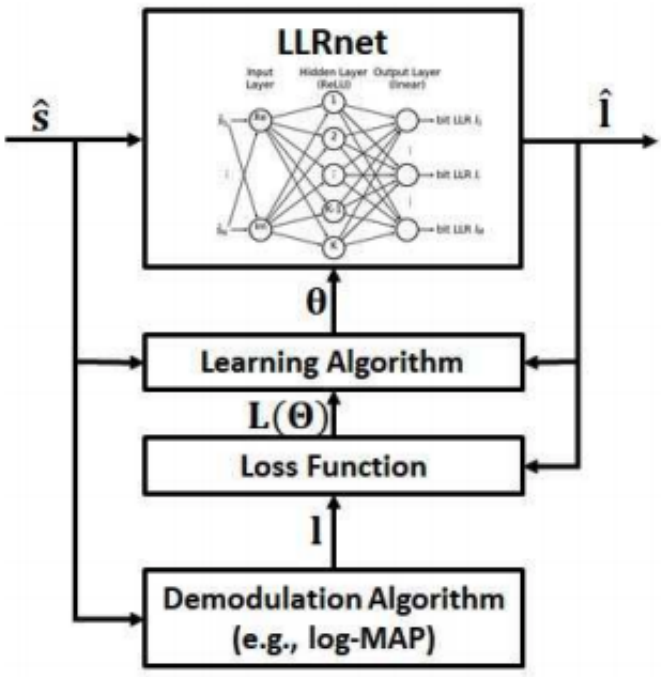

Figura 4.7: Diagrama de bloques del entrenamiento de la LLRNet [26].

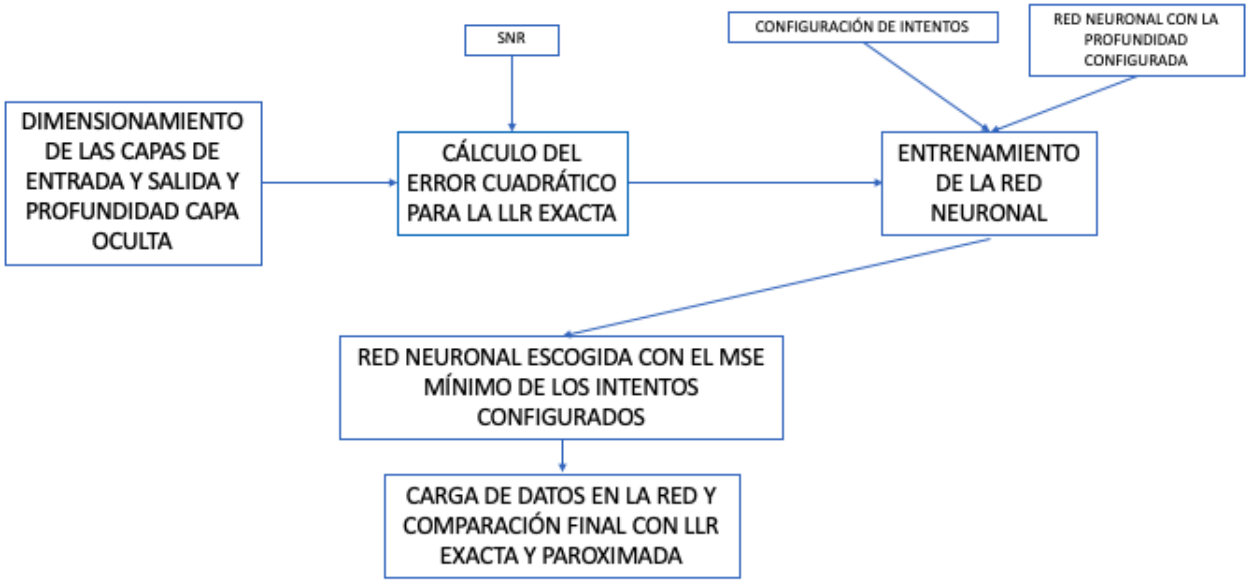

Figura 4.8: Diagrama Parte 2

Capítulo 5

Resultados de la Demo de Matlab y discusión de los resultados

5.1 Parte 1

En esta primera parte se van a realizar simulaciones, todas ellas independientes entre sí, donde se van a variar distintas variables de entrada para poder analizar el comportamiento de la red neuronal. Para ello se tomará siempre como referencia para cada simulación la curva correspondiente a la LLR Exacta, la cual será comparada con el resto de curvas (LLR-Net y LLR Aproximada).

Primeramente, hay que tener clara la diferencia de lo que se representa tanto en las Tablas como en las Figuras. Las Tablas muestran los datos de entrenamiento por parte de la red neuronal, como resultado del entrenamiento se ha querido mostrar para cada simulación: el MS LLR para cada SNR que corresponde al valor cuadrático medio para cada SNR y por otro lado los *Trials* o Pruebas donde se refleja el error (MSE) de la red neuronal ya entrenada, señalando en negrita la mejor prueba, que coincidirá con el menor error MSE. Por otro lado, las Figuras muestran los cálculos de LLR en el régimen permanente y servirán para la comparación de curvas de las distintas LLR.

5.1.1 Variación del orden de modulación (M)

Se va a proceder al entrenamiento de la red neuronal con los valores por defecto. La Tabla 5.1 corresponde a la misma simulación que la Figura 5.1. Para realizar esta simulación se ha utilizado los valores por defecto de la Demo con el orden de modulación de 16.

- Orden de Modulación= 16.

- Número de símbolos= 1e4.
- Profundidad de capa oculta= 8.
- Valores de SNR= [-5 0 5]. Estos valores son los valores que vienen por defecto en la Demo de Matlab y no se han variado para el resto de simulaciones.

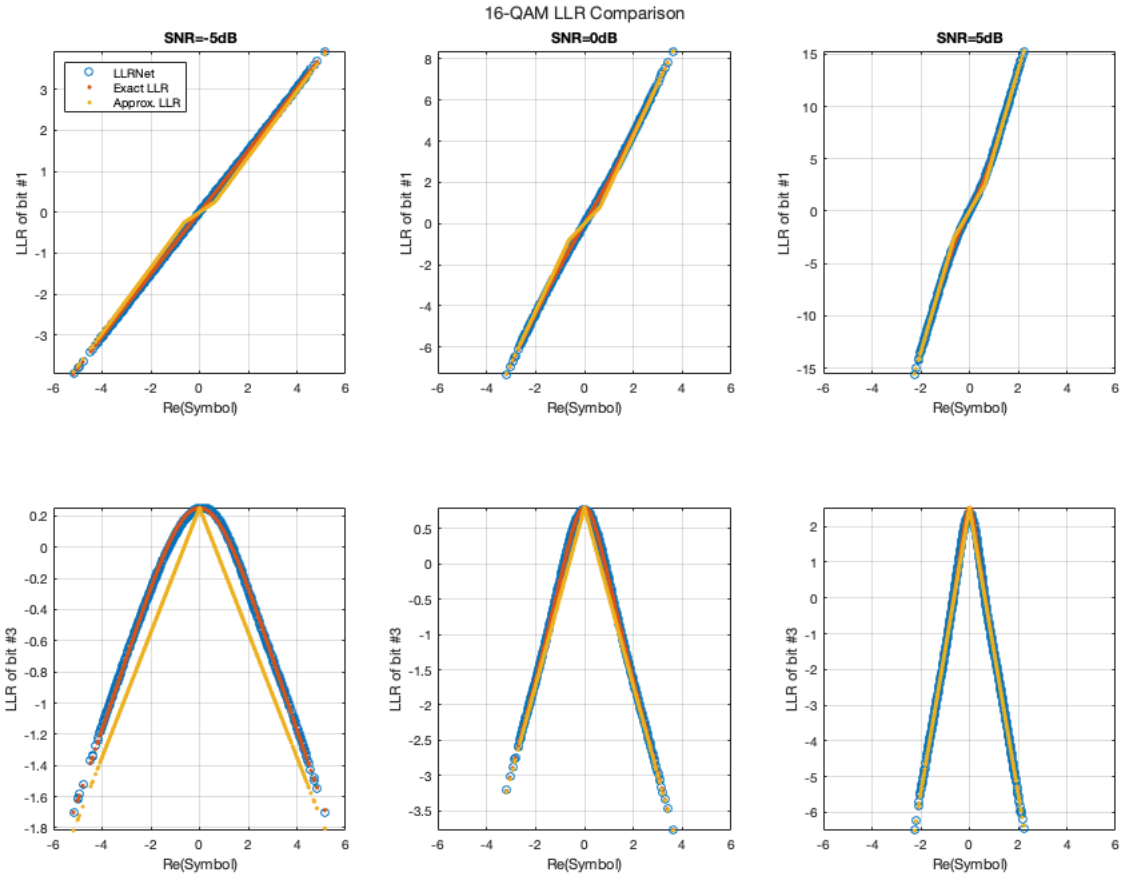

Figura 5.1: Resultados para 16QAM

Tabla 5.1: Tabla de resultados para 16QAM

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	4,43	6,86e-05	6,86e-05	**1,74e-06**	1,23e-04	2,90e-05
0	15,65	**4,09e-04**	1,04e-02	1,88e-03	1,88e-03	6,62e-04
5	59,64	5,10e-03	5,18e-03	**5,02e-03**	2,17e-02	2,18e-02

La Tabla 5.2 corresponde a la misma simulación que la Figura 5.2. Para realizar esta simulación se ha modificado el orden de modulación a 64.

- Orden de Modulación= 64.

- Número de símbolos= 1e4.
- Profundidad de capa oculta= 8.
- Valores de SNR= [-5 0 5].

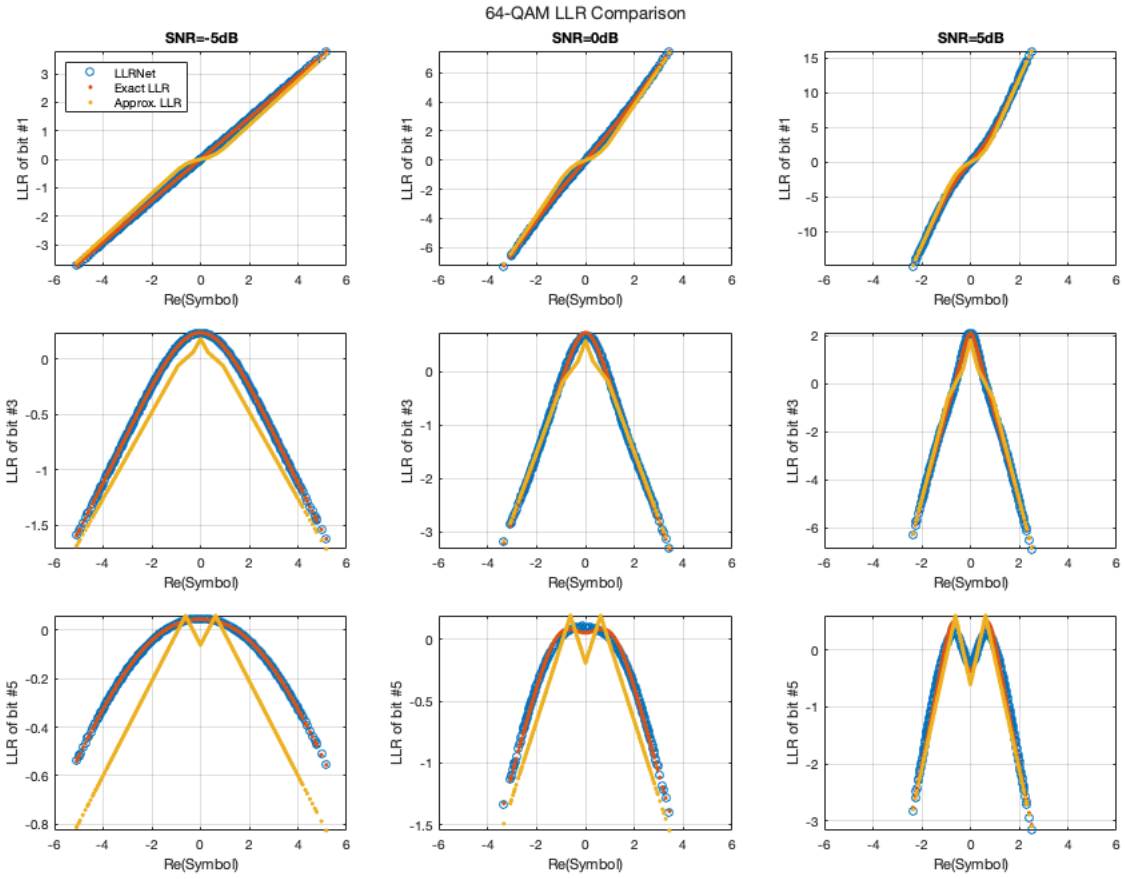

Figura 5.2: Resultados para 64QAM

Tabla 5.2: Tabla de resultados para 64QAM

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	2,89	**3.47e-06**	1,00e-05	2,30e-04	6,29e-06	2,35e-04
0	10.34	5,73e-04	3.04e-04	3.18e-04	3,02e-03	**4,79e-04**
5	37,11	2,94e-02	5,40e-02	9.08e-03	6,93e-03	**6,74e-03**

Para finalizar el apartado se repiten las simulaciones para el valor máximo de modulación posible en esta Demo. La Tabla 5.3 corresponde a la misma simulación que la Figura 5.3. Para realizar esta simulación se ha modificado el orden de modulación a 256.

- Orden de Modulación= 256.

- Número de símbolos= 1e4.
- Profundidad de capa oculta= 8.
- Valores de SNR= [-5 0 5].

La Tabla 5.3 corresponde a la Figura 5.3, en negrita se resalta la mejor prueba que corresponde al menor valor de error de todas las pruebas.

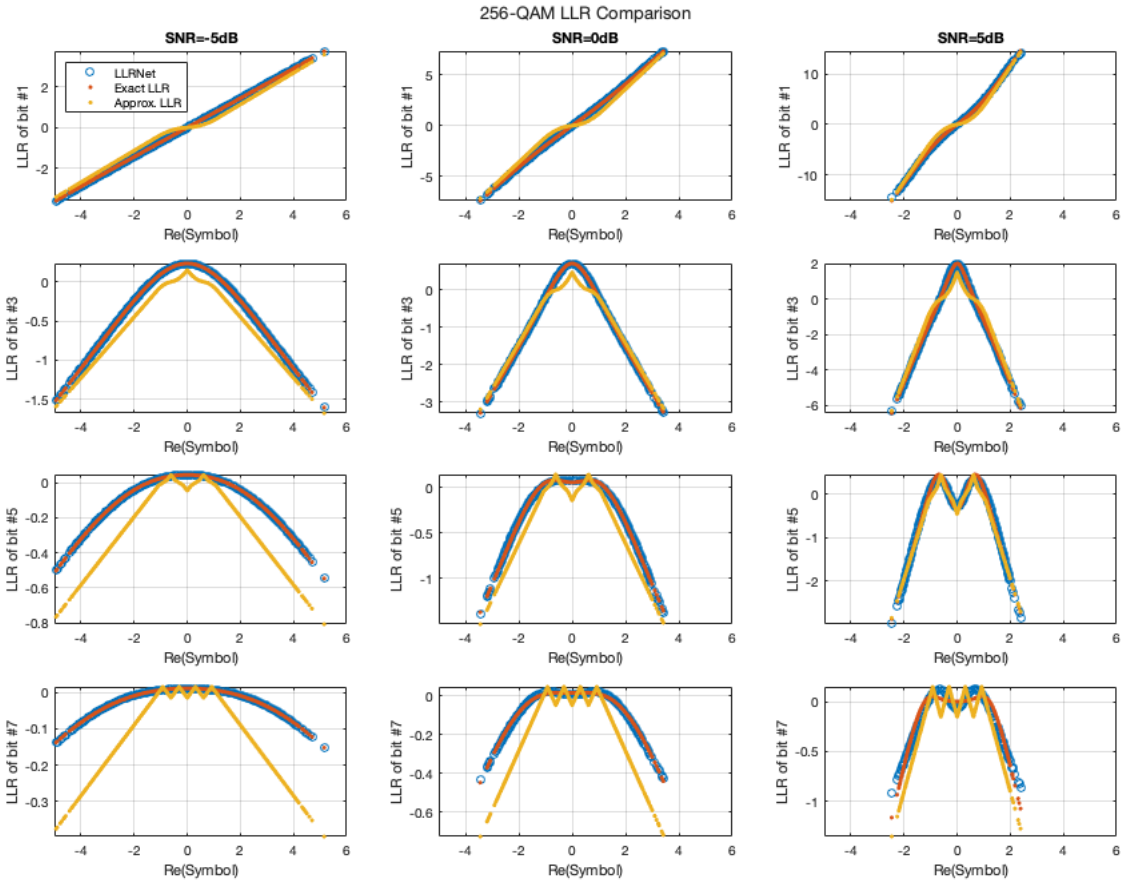

Figura 5.3: Resultados para 256QAM

Tabla 5.3: Tabla de resultados para 256QAM

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	1,77	2,63-06	1,38e-04	3,94e-06	1,38e-04	**1,07e-06**
0	5,47	1,83e-03	1,99e-04	**1,19e-04**	1,74e-04	1,74e-04
5	15,01	3,25e-03	**2,53e-03**	1,96e-02	3,97e-03	3,25e-03

5.1.2 Variación de la profundidad de la capa oculta

En este apartado se va a analizar el comportamiento del sistema al variar únicamente la profundidad de la capa oculta, de tal modo que para cada orden de modulación (16, 64 y 256), se han realizado simulaciones para una profundidad de 4 y de 16.

La Tabla 5.4 corresponde a la misma simulación que la Figura 5.4. Para realizar esta simulación se ha modificado la profundidad de la capa oculta a 4:

- Orden de Modulación= 16.
- Número de símbolos= 1e4.
- Profundidad de capa oculta= 4.
- Valores de SNR= [-5 0 5].

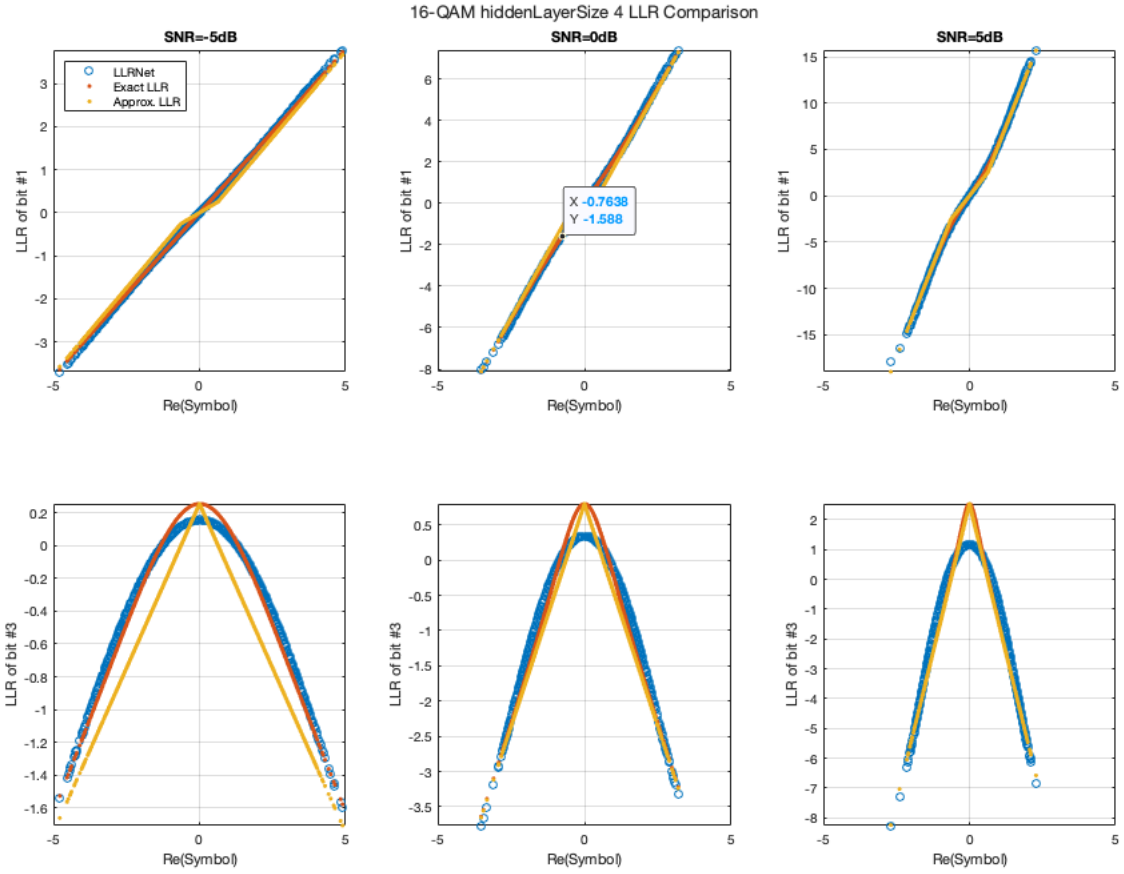

Figura 5.4: Resultados para 16QAM Capa Oculta 4

La Tabla 5.5 corresponde a la misma simulación que la Figura 5.5. Para realizar esta simulación se ha modificado la profundidad de la capa oculta a 16:

Tabla 5.4: Tabla de resultados para 16QAM Capa Oculta 4

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	4,35	**1,81e-03**	1,81e-03	2,09e-03	2,10e-03	2,08e-03
0	15,82	**2,74e-02**	2,82e-02	2,75e-02	5,98e-01	2,80e-02
5	58,68	**1,48e-01**	3,37e-01	1,48e-01	1,48e-01	1,48e-01

- Orden de Modulación= 16.
- Número de símbolos= 1e4.
- Profundidad de capa oculta= 16.
- Valores de SNR= [-5 0 5].

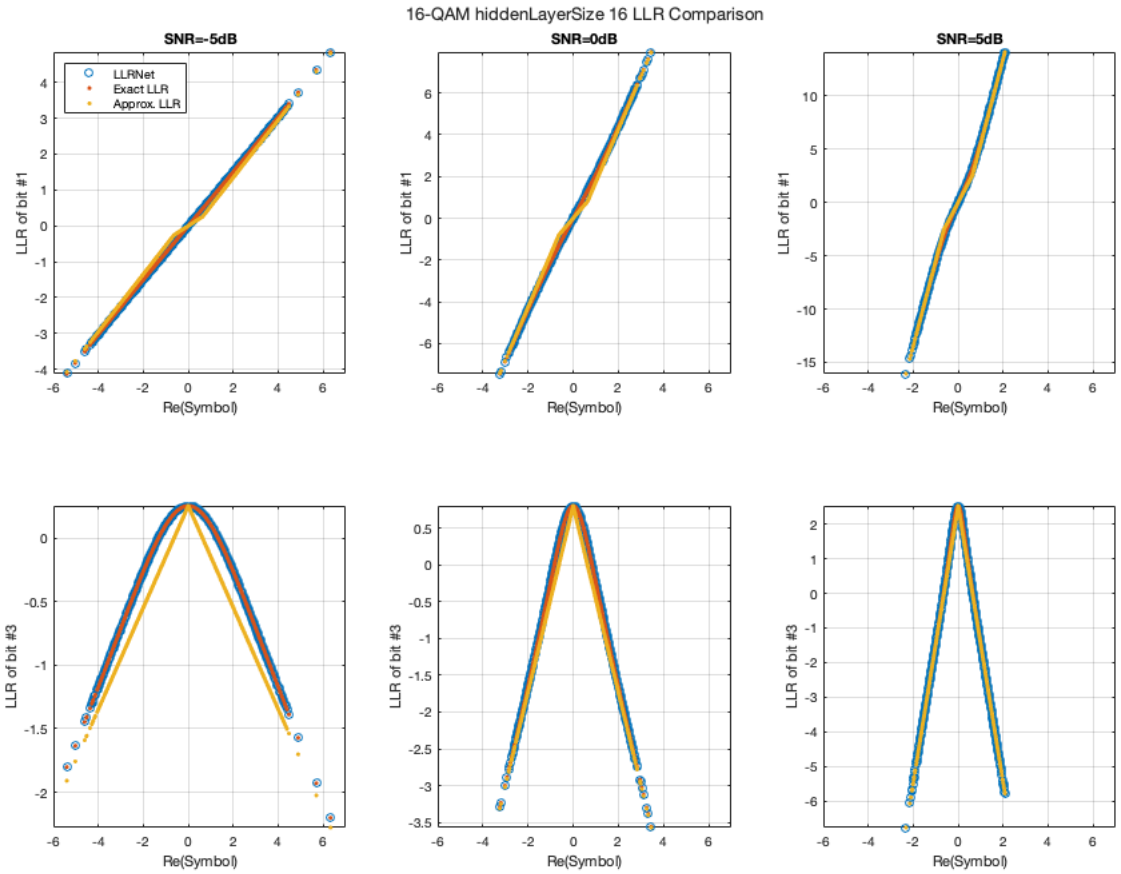

Figura 5.5: Resultados para 16QAM Capa Oculta 16

La Tabla 5.6 corresponde a la misma simulación que la Figura 5.6. Para realizar esta simulación se ha modificado la profundidad de la capa oculta a 4:

- Orden de Modulación= 64.

Tabla 5.5: Tabla de resultados para 16QAM Capa Oculta 16

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	4,46	3,58e-08	6,31e-08	**1,68e-08**	5,32e-07	1,23e-06
0	15,63	**3,98e-06**	2,25e-05	4,48e-06	**1,51e-06**	3,78e-06
5	59,69	**4,67e-05**	9.38e-05	8,41e-05	2,65e-04	8,30e-05

- Número de símbolos= 1e4.
- Profundidad de capa oculta= 4.
- Valores de SNR= [-5 0 5].

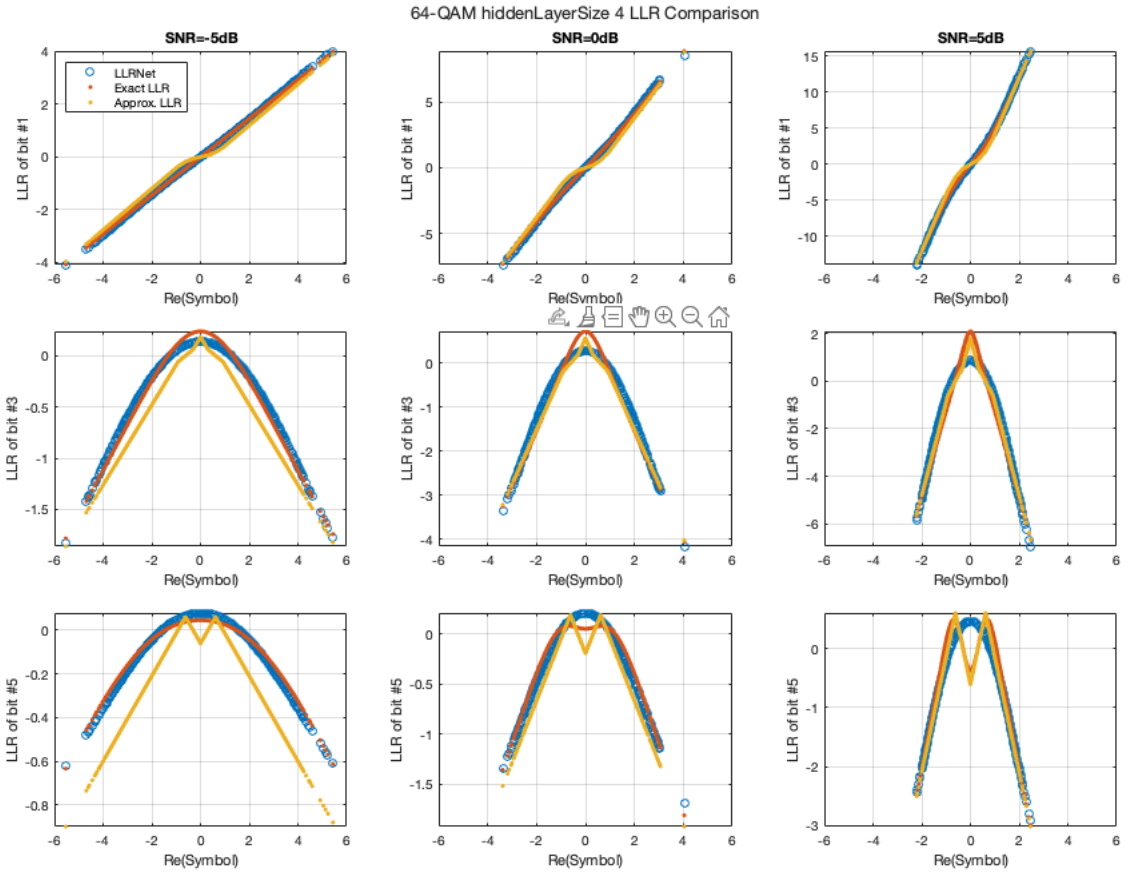

Figura 5.6: Resultados para 64QAM Capa Oculta 4

La Tabla 5.7 corresponde a la misma simulación que la Figura 5.7. Para realizar esta simulación se ha modificado la profundidad de la capa oculta a 16:

- Orden de Modulación= 64.
- Número de símbolos= 1e4.

Tabla 5.6: Tabla de resultados para 64QAM Capa Oculta 4

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	2,91	**1,50e-03**	1,06e-01	1,72e-03	1,06e-01	1,72e-03
0	10,35	1,94e-02	1,96e-02	1,96e-02	1,95e-02	**1,83e-02**
5	37,60	**1,09e-01**	1,09e-01	2,78e-01	1,09e-01	1,09e-01

- Profundidad de capa oculta= 16.
- Valores de SNR= [-5 0 5].

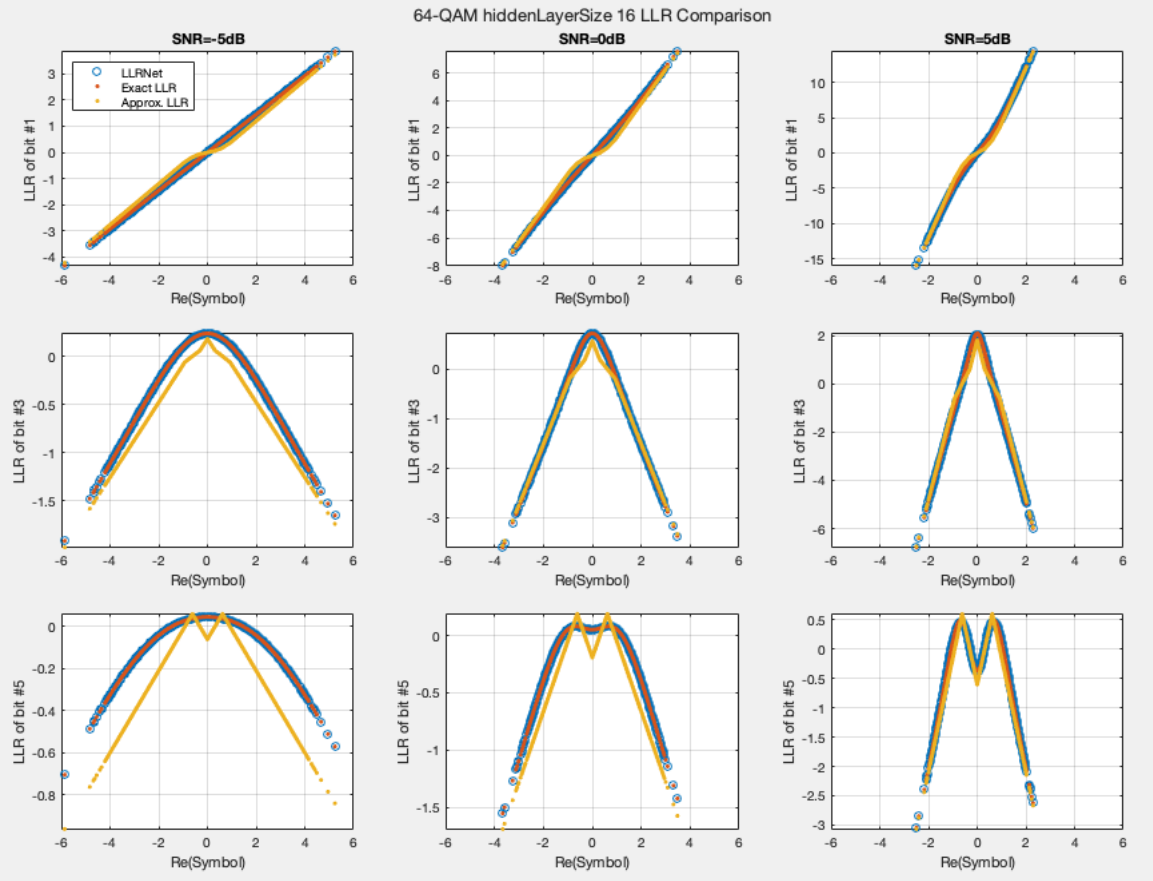

Figura 5.7: Resultados para 64QAM Capa Oculta 16

Para finalizar la Tabla 5.8 corresponde a la misma simulación que la Figura 5.8. Para realizar esta simulación se ha modificado la profundidad de la capa oculta a 4:

- Orden de Modulación= 256.
- Número de símbolos= 1e4.
- Profundidad de capa oculta= 4.
- Valores de SNR= [-5 0 5].

Tabla 5.7: Tabla de resultados para 64QAM Capa Oculta 16

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	2,92	6,32e-08	1,00e-07	1,28e-07	**1,59e-08**	1,31e-07
0	10,34	3,76e-06	1,09e-06	**8,57e-07**	1,25e-06	2,01e-06
5	36,83	2,03e-04	1,83e-04	9,80e-05	**4,02e-05**	1,28e-04

Tabla 5.8: Tabla de resultados para 256QAM Capa Oculta 4

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	1,75	8,80e-04	8,81e-04	**8,31e-04**	6,36e-02	8,80e-04
0	5,47	1,04e-02	9,86e-03	**9,86e-03**	9,86e-03	9,86e-03
5	14,94	**6,84e-02**	6,84e-02	6,84e-02	6,84e-02	6,84e-02

Se finaliza con la Tabla 5.9 corresponde a la misma simulación que la Figura 5.9. Para realizar esta simulación se ha modificado la profundidad de la capa oculta a 16:

- Orden de Modulación= 256.
- Número de símbolos= 1e4.
- Profundidad de capa oculta= 16.
- Valores de SNR= [-5 0 5].

Tabla 5.9: Tabla de resultados para 256QAM Capa Oculta 16

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	1,76	3,02e-08	**6,68e-09**	1,05e-08	2,85e-08	2,36e-08
0	5,40	1,59e-06	5,47e-07	**4,10e-07**	6,74e-07	4,88e-07
5	14,89	4,42e-05	7,26e-05	**3,54e-05**	4,11e-05	5,38e-05

5.1.3 Variación del número de símbolos

Para concluir la Parte 1 se dispone a variar el número de símbolos. Las simulaciones se han realizado para un único orden de modulación, 16 y para una profundidad también constante, 16.

- Orden de Modulación= 16.
- Número de símbolos= 1e3.

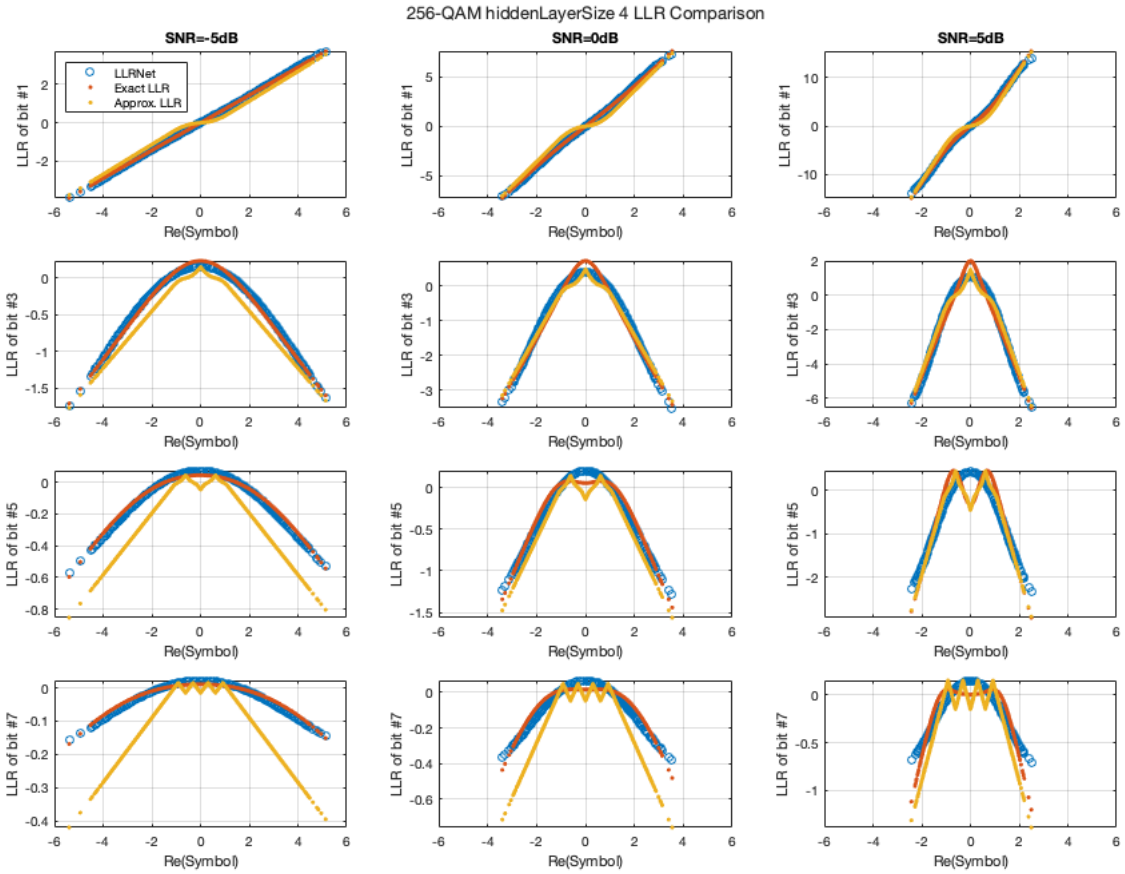

Figura 5.8: Resultados para 256QAM Capa Oculta 4

- Profundidad de capa oculta= 16.
- Valores de SNR= [-5 0 5].

La Tabla 5.10 corresponde a la Figura 5.10 y de la misma forma, en negrita se resalta el valor con menor error de todas as pruebas realizadas.

Tabla 5.10: Tabla de resultados para 1000 símbolos

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	4,53	9,40e-07	**6,05e-08**	8,40e-08	3,03e-07	6,51e-08
0	14,98	4,48e-06	**5,41e-07**	1,78e-06	4,34e-05	4,18e-06
5	56,62	6,87e-04	**1,02e-04**	2,63e-04	3,75e-04	3,77e-04

Simulaciones realizadas para 1e5 símbolos:

- Orden de Modulación= 16.
- Número de símbolos= 1e5.

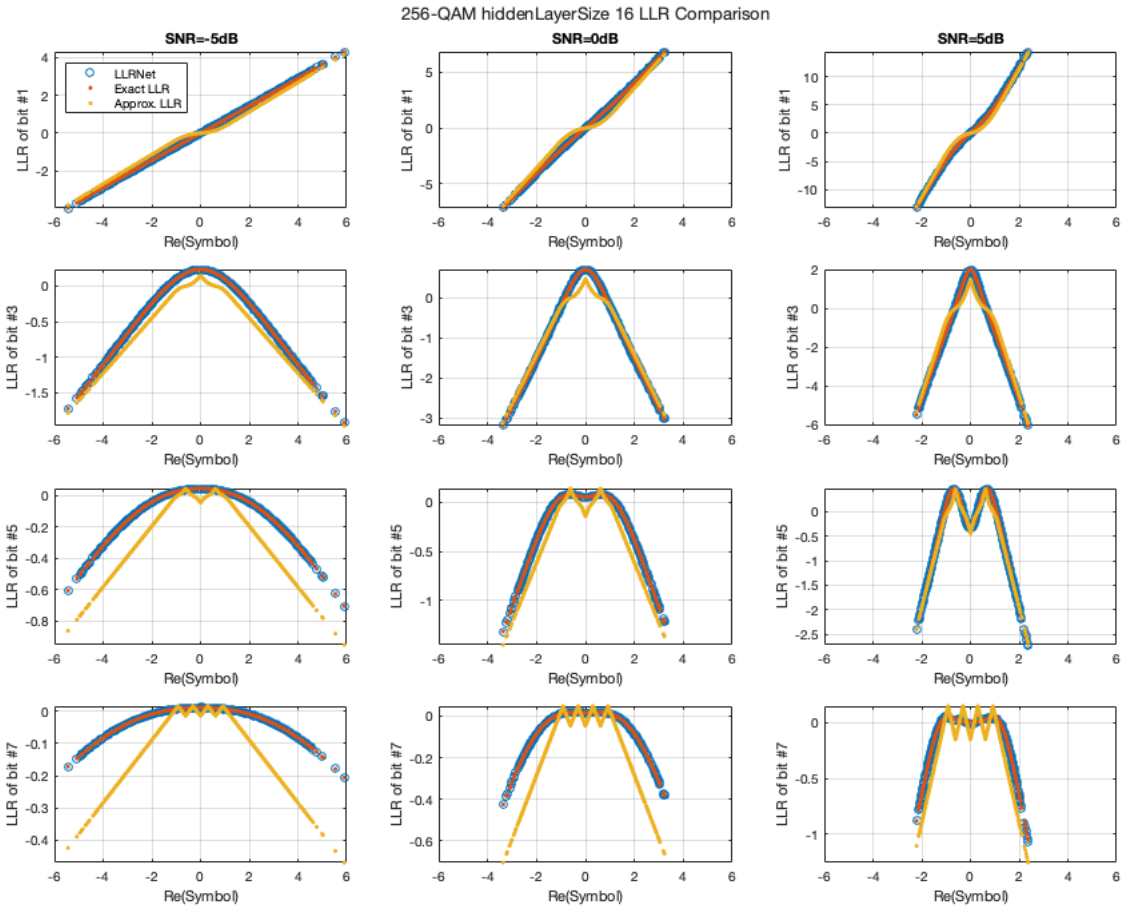

Figura 5.9: Resultados para 256QAM Capa Oculta 16

- Profundidad de capa oculta= 16.

- Valores de SNR= [-5 0 5].

La Tabla 5.11 corresponde a la Figura 5.11.

Tabla 5.11: Tabla de resultados para 100000 símbolos

SNR	MS LLR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
-5	4,43	**3,07e-08**	2,01e-07	3,18e-07	3,07e-07	7,21e-07
0	15,74	1,21e-06	**6,43e-07**	1,26e-05	3,48e-06	1,25e-05
5	59,51	1,29e-04	8,12e-05	**4,89e-05**	2,18e-04	2,49e-04

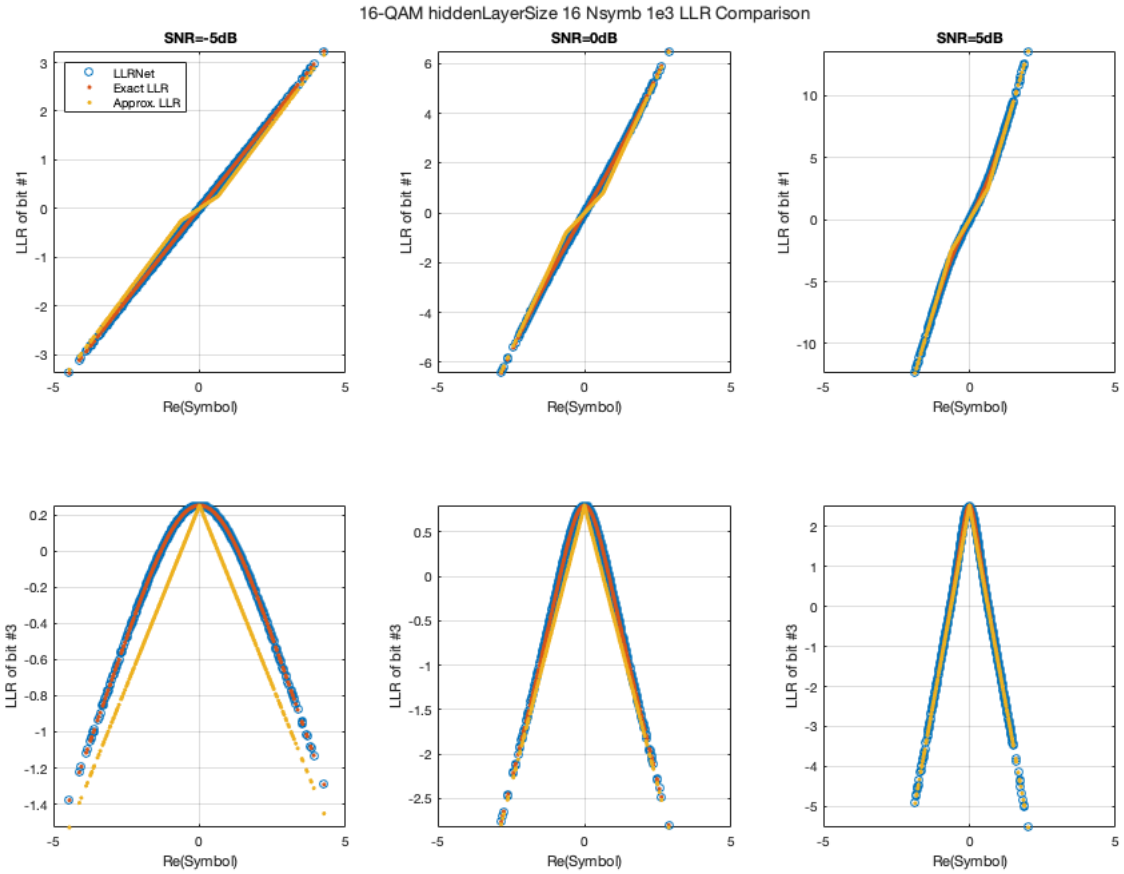

Figura 5.10: Resultados para 1000 símbolos

5.2 Parte 2. *DVB-S.2 Packet Error Rate*

En este Parte se tratará de analizar las simulaciones correspondientes con la segunda parte del código de la Demo, en la cual se trata el PER. Para evaluar el PER se variarán ciertos componentes de entrada para entrenar una red neurona, donde además se simulará la cadena de conexión para finalizar con la estimación del PER .

5.2.1 Variación de la profundidad de la capa oculta

Para este apartado se procederá a variar únicamente la profundidad de la capa oculta.

Para la Figura 5.12 se ha procedido a simular con los siguientes parámetros:

- Orden de Modulación= 16
- Modulación: APSK
- Número de símbolos= 1e4.

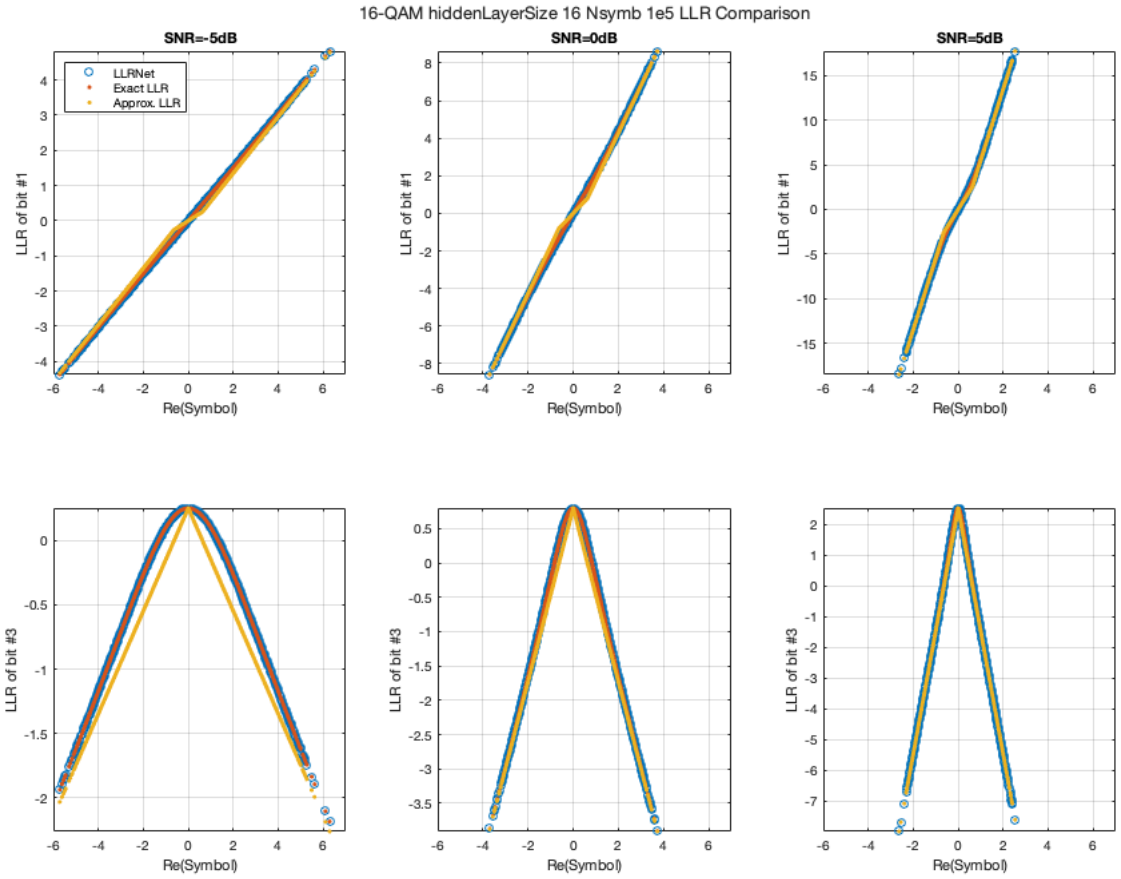

Figura 5.11: Resultados para 100000 símbolos

- Profundidad de capa oculta= 32.
- Tasa de Codificación= 2/3.
- Número de errores= 200;

Simulación para profundidad de capa oculta 64, Figura 5.13:

- Orden de Modulación= 16
- Modulación: APSK
- Número de símbolos= 1e4.
- Profundidad de capa oculta= 64.
- Tasa de Codificación= 2/3.
- Número de errores= 200;

Para la Figura 5.14 se ha procedido a simular con los siguientes parámetros:

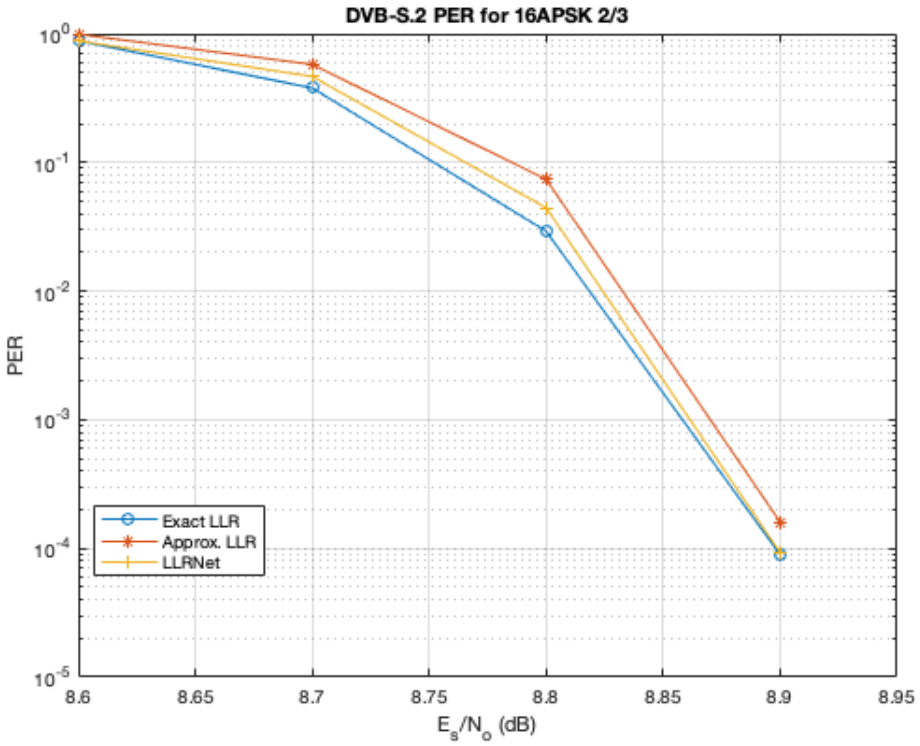

Figura 5.12: Resultados PER 16 APSK 2/3 y 1e4 símbolos para profundidad L=32

- Orden de Modulación= 16
- Modulación: APSK
- Número de símbolos= 1e4.
- Profundidad de capa oculta= 128.
- Tasa de Codificación= 2/3.
- Número de errores= 200;

5.2.2 Variación del número de símbolos

Para este apartado se estudiará el comportamiento cuando se varía el número de símbolos. Figura 5.15:

- Orden de Modulación= 16
- Modulación: APSK
- Número de símbolos= 1e2.

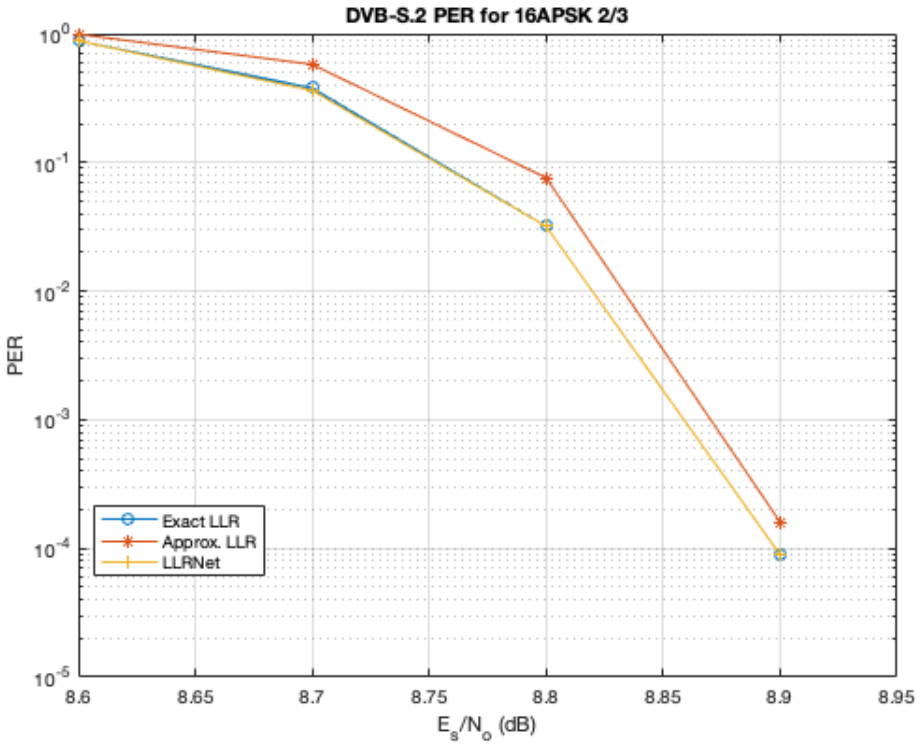

Figura 5.13: Resultados PER 16 APSK 2/3 y 1e4 símbolos para profundidad L=64

- Profundidad de capa oculta= 64.
- Tasa de Codificación= 2/3.
- Número de errores= 200;

Figura 5.16:

- Orden de Modulación= 16
- Modulación: APSK
- Número de símbolos= 1e3.
- Profundidad de capa oculta= 64.
- Tasa de Codificación= 2/3.
- Número de errores= 200;

Figura 5.17:

- Orden de Modulación= 16
- Modulación: APSK

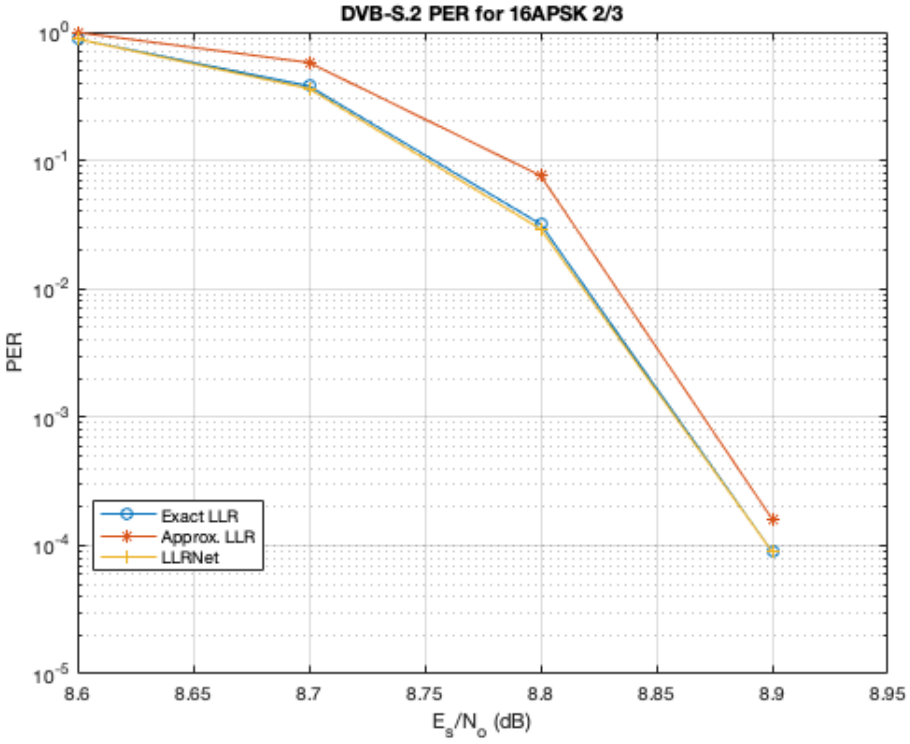

Figura 5.14: Resultados PER 16 APSK 2/3 y 1e4 símbolos para profundidad L=128

- Número de símbolos= 1e5.
- Profundidad de capa oculta= 64.
- Tasa de Codificación= 2/3.
- Número de errores= 200;

5.3 Análisis Parte 1

5.3.1 Análisis de la Variación de la Modulación

Un factor común de este apartado que se puede observar en las Tablas 5.1, 5.2 y 5.3 es que el valor de la MSE aumente conforme más SNR tenga el sistema. Cabe destacar una vez más que las Tablas anteriormente mencionadas corresponden a pruebas sucesivas e independientes de entrenamiento donde el sistema se queda con la prueba con menor error. Es un hecho que ya se esperaba ya que el valor de partida de la LLR que se quiere calcular es más alto. Este factor se repetirá en el resto de gráficas de la Parte 1.

Por otro lado queda reflejado en las Tablas 5.1, 5.2 y 5.3 que el valor de error MS LLR es menor conforme se aumenta el orden de modulación, esto es debido a que las LLR se calculan por bit,

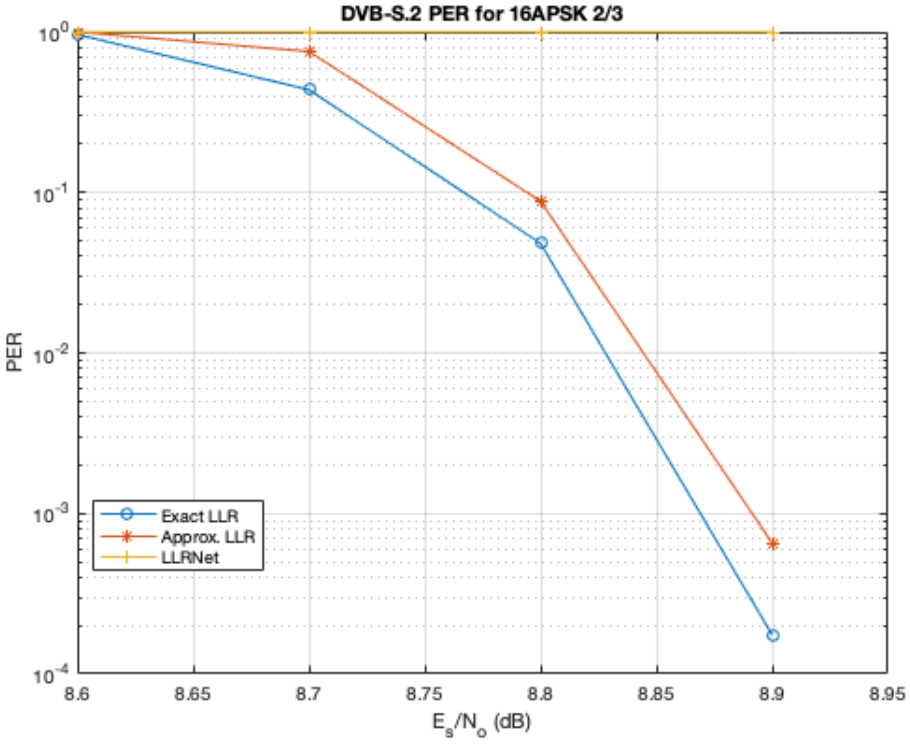

Figura 5.15: Resultados PER 16 APSK 2/3, L=64 y 1e2 símbolos

pero se trabaja sobre símbolos. A mayor orden, mayor número de bits por símbolo lo que significa que, al calcular la LLR bit a bit, el valor cuadrático medio de la LLR va a ser menor, es decir, habrá menos certidumbre sobre cada bit de dicho símbolo y coincide con lo esperado. También hay que destacar que a mayor SNR la MS LLR también se hace mayor lo cual compensa el aumento de la MSE comentado anteriormente.

Al estudiar la Figura 5.1 se puede observar como la curva de color rojo (LLR Exacta) y la señal amarilla (LLR-Net) son prácticamente idénticas en todas las gráficas lo que demuestra la efectividad y el gran rendimiento de la LLR-Net entrenada. Para el caso de la Figura 5.1 se puede ver que las curvas de la LLR Aproximada puede dar lugar a pensar que su utilización es apropiada, pero en la gráfica de abajo a la izquierda, en forma de pico, se ve una gran diferencia con respecto al resto de curvas, donde llama la atención que casi no se distingue la curva roja y azul lo que demuestra la fiabilidad y el mínimo error de la LLR-Net.

Al aumentar el orden de modulación se producen variaciones, como se puede ver en la Figura 5.2 para una modulación de 64 QAM. Para el bit 1 se observa una diferencia mínima entre las 3 curvas para todos los tipos de SNR, incluso para el bit 3 la diferencia es más visible con respecto a la LLR Aproximada, pero sigue siendo una diferencia leve. Sin embargo para el bit 5 se puede ver como la diferencia de la LLR Aproximada hace que haga poco deseable su utilización en esta modulación. Sin duda, el hecho remarcable de la Figura 5.2 es la similitud entre la LLR-Net y

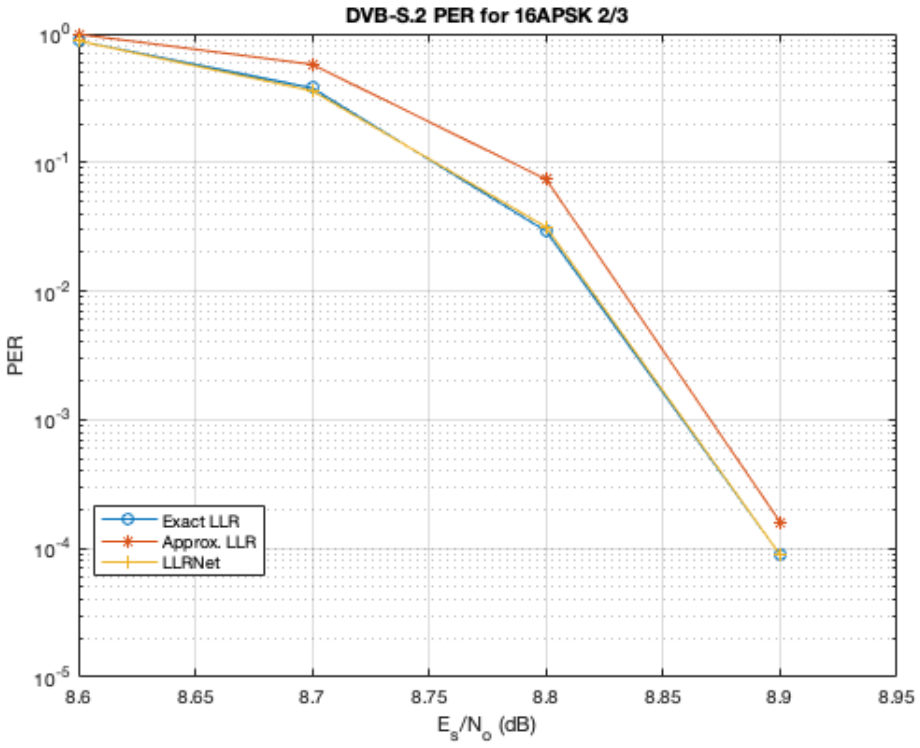

Figura 5.16: Resultados PER 16 APSK 2/3, L=64 y 1e3 símbolos

la LLR Exacta, casi idéntica en todas las gráficas, lo que refleja lo idóneo de utilizar una red neuronal, con un error mínimo y reduciendo el gran consumo de recursos.

Analizando la Figura 5.3 para una modulación de 256 QAM, se observa a simple vista la gran diferencia entre la curva amarilla (LLR Aproximada) con respecto a las otras dos, por lo que de entrada será menos apropiada para su uso. Por otro lado, se puede ver una vez más como las curvas roja y azul, que corresponden a la LLR Exacta y LLR-Net, respectivamente, son prácticamente idénticas, , lo que refleja lo idóneode utilizar una red neuronal, con un error mínimo y reduciendo el gran consumo de recursos.

El hecho de que las curvas de LLR Exacta y LLR-Net coincidan ocurre durante el funcionamiento del sistema en régimen permanente pero se encuentran algunos inconvenientes también, ya que hay que entrenar la red y se debe tener una red distinta para cada condición del canal y para cada juego de parámetros.

5.3.2 Análisis de la variación de profundidad de la capa oculta

Tanto al aumentar como al disminuir la profundidad de la capa oculta se han encontrado una serie de diferencias que han llevado a determinar que, a menor profundidad de capa oculta, el MSE es mucho mayor, lo cual es un resultado que a priori se esperaba obtener ya que al tener

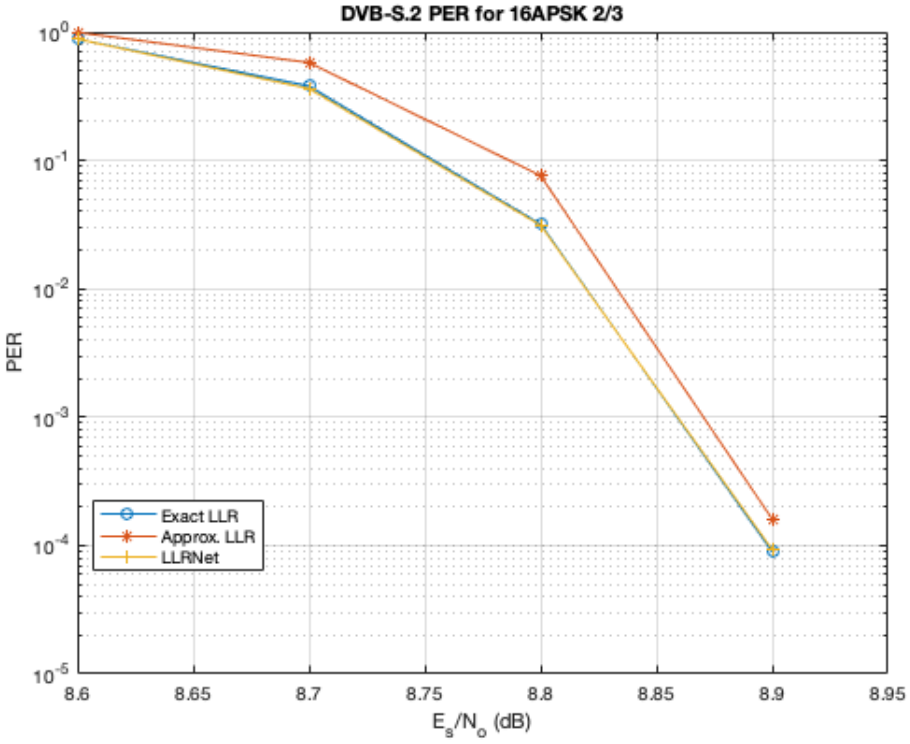

Figura 5.17: Resultados PER 16 APSK 2/3, L=64 y 1e5 símbolos

menos profundidad el entrenamiento de la red será menor puesto que no tiene recursos físicos para aprender más. De otra forma, si se incrementa la profundidad de la capa oculta la red tendrá un entrenamiento y capacidad mayores que le permitirá obtener resultados de manera más precisa. Esto queda reflejado en las Tablas 5.4, 5.5, 5.6, 5.7, 5.8 y 5.9.

Si se observan ahora las Figuras correspondientes a este apartado, en la Figura 5.4 se ve como para el bit 1 las curvas presentan una semejanza considerable pero, para el bit 3, las diferencias tanto de la LLR Aproximada como de la LLR-Net con respecto a la LLR Exacta son notables lo cual hará que resultase menos adecuado su uso. Si se estudia la Figura 5.5, se puede ver que la LLR-Net es muy similar a la LLR Exacta, incluso se puede comparar también con la Figura 5.1 donde se analizaba la comparativa entre funciones para 16 QAM y 8 de profundidad de capa oculta y ambas Figuras son también muy similares pero se puede ver como para la Figura 5.5 hay un poco más de similitud entre señales roja y azul, que es una vez más lo que se esperaba a priori al tener menos MSE al tener más profundidad. Este hecho se puede contrastar viendo las Tablas 5.1 y 5.5 donde queda reflejado la reducción del MSE conforme aumentamos la profundidad de la capa oculta (aplicable para el resto de modulaciones). De la Figura 5.4 se puede llegar a la conclusión de que para profundidades de capa oculta pequeñas y 16 QAM, es más conveniente usar la LLR Aproximada por el comportamiento que presenta en las gráficas debido al MSE que la LLR-Net acarrea y que se ha comentado anteriormente.

Al estudiar las variaciones producidas por la disminución y el aumento de la profundidad de la capa oculta para 64 y 256 QAM en las Figuras 5.6 y 5.7 ,se puede llegar a la misma conclusión que se ha llegado para 16 QAM ,pero con una diferencia ,y es que para el bit 5 y 7 la LLR Aproximada pierde efectividad y la diferencia entre esta y la LLR Exacta se hace notoria. De igual manera se puede comparar las Tablas desde la 5.4 hasta la 5.9 con respecto a las Tablas de la 5.1 hasta la 5.3 para observar como a mayor profundidad de capa oculta la red neuronal presentará un MSE menor.

5.3.3 Análisis de la variación del número de símbolos para 16QAM

El resultado obtenido ha vuelto a ser el esperado y es que al fijarse en las Figuras 5.10 y 5.11 se observa que al aumentar el número de símbolos la red neuronal tendrá un conocimiento y un entrenamiento más apropiado el cual queda reflejado en la similitud que presenta las gráficas en las Figuras 5.10 y 5.11.

5.4 Análisis Parte 2

5.4.1 Análisis de la variación de la profundidad de la capa oculta

Los resultados de este apartado se pueden visualizar en las Figuras 5.12, 5.13 y 5.14. Aunque no se encuentre apenas diferencias entre la Figura 5.13 y 5.14 con profundidad de la capa oculta de 32 y 64, respectivamente. La disparidad entre las Figuras 5.13 y 5.14 (profundidad de capa oculta 128) y la Figura 5.12 es evidente y el resultado obtenido es lógico ya que cuanto mas entrenada está una red neuronal menos MSE va a tener y más se va a asemejar a la LLR Exacta, lo cual se puede comprobar en la Figura 5.12 que es bastante notoria la diferencia entre la LLR Exacta y la LLR-Net. También cabe destacar el gran error que presenta la LLR Aproximada con respecto a la LLR Exacta y hace que sea menos competitivo su uso desde este punto de vista.

5.4.2 Análisis de la variación del número de símbolos

Por último, se han realizado simulaciones para comparar y analizar los resultados obtenidos al variar el número de símbolos usados en el entrenamiento, obteniendo un error grande para 1e2 símbolos de (10^1) lo que determina que se tenga que entrenar con un número de símbolos mínimo de 1e3, pero sí se puede afirmar que a mayor número de símbolos se obtendrá un resultado con menor error, porque la red neuronal estará mejor entrenada.

A continuación se presentarán las Conclusiones finales que se han llegado fruto de las simu-

laciones y el análisis realizado en esta parte así como las Líneas Futuras donde se aportarán sugerencias para poder completar este proyecto o posibles variaciones.

Capítulo 6

Conclusiones y líneas futuras

6.1 Conclusiones

Se han obtenido numerosas conclusiones después de realizar las diversas simulaciones, variando los distintos parámetros:

- Al variar el índice de modulación (M) se ha llegado a la conclusión de que a mayor índice de modulación se obtiene un MS LLR menor debido a que las LLR se calculan por bit, pero se trabajan sobre símbolos, y esto lleva a que, a mayor orden, mayor número de bits por símbolo por lo que al calcular la LLR bit a bit, el valor cuadrático medio va a ser menor. Sin embargo el MSE obtenido aumenta al aumentar la SNR esto es debido a que el valor de partida de la LLR que se quiere calcular es más alto. Por otro lado se ha comprobado que, a pesar de aumentar el orden de modulación, los valores de las MSE para cada SNR son similares en todos los órdenes de modulación, como se puede observar en las Tablas 5.1, 5.2 y 5.3. Debido a ello se llega a la conclusión de que el orden de modulación no afecta al error de la red neuronal pero sí que hace menos competitivo el uso de la LLR Aproximada desde ese punto de vista.

- Tras realizar variaciones en la profundidad de la capa oculta se ha podido confirmar que es una variable determinante en el rendimiento de la red, ya que a menor profundidad peor será el resultado final.

- Por último se ha modificado el número de símbolos con los que entrenar a la red. Se ha llegado a la confirmación de que una red neuronal entrenada con menos símbolos es más imprecisa que otra red la cual haya sido entrenada con un número de símbolos mayor.

  Esta afirmación se puede hacer ya que no se ha experimentado sobreentrenamiento al aumentar el número de símbolos, pero es un suceso que puede llegar a darse.

La conclusión fundamental a la que se ha llegado al realizar este trabajo es que las RNA son viables, fiables, fácilmente aplicables, baratas y sin duda alguna estarán más presentes en el futuro tecnológico de la sociedad.

Con el entrenamiento apropiado de estas redes la sociedad será capaz de resolver complejos problemas reduciendo considerablemente los gastos en recursos y en tiempo operacional.

A su vez las redes neuronales también presentan inconvenientes que se han podido experimentar al elaborar este trabajo. Y es que la red ha de ser entrenada, y esos datos obtenidos almacenados, para cada situación el canal y juego de parámetros, siendo imprescindible monitorizar la configuración y el ruido constantemente para poder utilizar una red u otra. Todo esto conlleva también un consumo de recursos que en comparación se deberían estar dispuestos a asumir.

6.2 Líneas futuras

Como líneas futuras del presente TFG que han surgido como resultado del trabajo realizado se han presentado algunas complicaciones a la hora de simular sobre todo la Parte 2 de la Demo ya que al intentar cambiar de modulación o de tasa de código las simulaciones aparecían vacías, seguramente por que estaba mal definido el rango de trabajo de la Es/No, lo que lo convierte en un trabajo manual y muy poco práctico, por lo que para este punto se podría dar alguna solución creando alguna función de barrido y ajuste del eje y esto se podrá abordar en forma en otro TFG.

Por otro lado, también se podría retocar el código de la Parte 1 de la Demo para poder simular otras modulaciones y comparar el funcionamiento de las RNA en esos casos.

Bibliografía

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[2] H. Cruickshank, M. P. Howarth, S. Iyengar, Z. Sun, and L. Claverotte. Securing multicast in dvb-rcs satellite systems. *IEEE Wireless Communications*, 12(5):38–45, 2005.

[3] Desconocido. La historia de las redes neuronales y la ia: Parte 3. *https://blog.techdata.com/ts/latam/la-historia-de-las-redes-neuronales-y-la-ia-parte-3*, 2018.

[4] M. P. Díaz. Fundamentos físicos de calidad de imagen en medicina nuclear. métodos para su valoración. *Revista Medicina Nuclear, Alasbimn Journal*, 9(35):1–15, 2007.

[5] E. ETSI. Digital video broadcasting (dvb); interaction channel for satellite distribution systems. *ETSI EN*, 301(790):V1, 2005.

[6] E. ETSI. Digital video broadcasting (dvb); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications. *Part II: S2-Extensions (DVB-S2X)*, pages 22–27, 2005.

[7] V. M. Fernández Solórzano, J. M. Pérez Llano, et al. Codificador ldpc e interleaver para dvb-s2. 2006.

[8] L. M. Guerra Hidalgo. Análisis y simulación del estándar dvb-s2 de televisión satélital usando varias modulaciones para los laboratorios de telecomunicaciones. B.S. thesis, 2017.

[9] D. F. Guerra Mina. Desarrollo de un programa que permita calcular los parámetros de transmisión forward en enlaces satelites utilizando el estándar dvb-s2 (digital video broadcasting by satelite 2). B.S. thesis, QUITO/EPN/2007, 2007.

[10] P. F. Guridi. Flujo de transporte mpeg-2.

[11] W. Hasperué. *Mapas auto-organizativos dinámicos.* PhD thesis, Universidad Nacional de La Plata, 2005.

[12] A. hombros de gigantes. Tema 8. redes neuronales. *https://iplogger.org/2sc6f6*, page 1, 2015.

[13] S. Landeros-Ayala, S. A. Chávez-Cárdenas, and J. C. González-Sánchez. Análisis de la eficiencia de los estándares de transmisión de televisión digital por satélite en las bandas ku y ka. *Ingeniería, investigación y tecnología*, 14(3):335–353, 2013.

[14] P. Larranaga, I. Inza, and A. Moujahid. Tema 8. redes neuronales. *Redes Neuronales, U. del P. Vasco*, 12:17, 1997.

[15] M. A. C. Maher, S. P. Deweerth, M. A. Mahowald, and C. A. Mead. Implementing neural architectures using analog vlsi circuits. *IEEE Transactions on circuits and systems*, 36(5):643–652, 1989.

[16] F. Mateo Jiménez. *Redes neuronales y preprocesado de variables para modelos y sensores en bioingeniería.* PhD thesis, Universitat Politècnica de València, 2012.

[17] D. J. Matich. Redes neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*, 41, 2001.

[18] Matlab. "training and testing a neural network for llr estimation". *https://es.mathworks.com/help/comm/ug/training-and-testing-a-neural-network-for-llr-estimation.html*, page 1, En línea.

[19] W. McCulloch and W. Pitts. Un cálculo lógico de la inminente idea de la actividad nerviosa. *Publicado en el boletín de matemática biofísica*, 5(115.133), 1943.

[20] A. Morera Munt and J. T. Alcalá Nalvaiz. Introducción a los modelos de redes neuronales artificiales el perceptrón simple y multicapa, 2018.

[21] X. B. Olabe. Redes neuronales artificiales y sus aplicaciones. *Publicaciones de la Escuela de Ingenieros*, 1998.

[22] U. Reimers. Digital video broadcasting. *IEEE communications Magazine*, 36(6):104–110, 1998.

[23] M. Rubiolo. Desarrollo de nuevos modelos y algoritmos basados en redes neuronales para tareas de minera de datos. 2014.

[24] L.-A. Salvador, S. A. Chávez-Cárdenas, and J. C. González-Sánchez. Análisis de la eficiencia de los estándares de transmisión de televisión digital por satélite en las bandas ku y ka. *Ingeniería, Investigación y Tecnología*, 14(3):335–353, 2013.

[25] Serrotho. La historia de las redes neuronales y la ia. 13/06/2018.

[26] J. H. . O. Shental. machine llrning": Learning to softly demodulate". *https://arxiv.org/pdf/1907.01512.pdf*, page 7, 2020.

[27] R. A. O. Vega. *Red de base radial: nueva aproximación en menos parámetros*. PhD thesis, 2014.

[28] Wikipedia. Conferencia de dartmouth — wikipedia, la enciclopedia libre, 2020. [Internet; descargado 29-septiembre-2021].

[29] Wikipedia. Karl lashley — wikipedia, la enciclopedia libre, 2021. [Internet; descargado 29-septiembre-2021].

[30] Wikipedia. Perceptrón — wikipedia, la enciclopedia libre, 2021. [Internet; descargado 29-septiembre-2021].

Anexo

DEMO MATLAB

https://es.mathworks.com/help/comm/ug/training-and-testing-a-neural-network-for-llr-estimation.html

PARTE 1

```
M = 16;
k = log2(M);
SNRValues = -5:5:5;
numSymbols = 1e4;
numSNRValues = length(SNRValues);
symOrder = llrnetQAMSymbolMapping(M);


const = qammod(0:15,M,symOrder,'UnitAveragePower',1);
maxConstReal = max(real(const));
maxConstImag = max(imag(const));


numBits = numSymbols*k;
exactLLR = zeros(numBits,numSNRValues)
approxLLR = zeros(numBits,numSNRValues);
rxSym = zeros(numSymbols,numSNRValues);
for snrIdx = 1:numSNRValues
SNR = SNRValues(snrIdx);
noiseVariance = 10POW(-SNR/10);
sigma = sqrt(noiseVariance);


maxReal = maxConstReal + 3*sigma;
minReal = -maxReal;
maxImag = maxConstImag + 3*sigma;
minImag = -maxImag;
```

```
r = (rand(numSymbols,1)*(maxReal-minReal)+minReal) + ...
1i*(rand(numSymbols,1)*(maxImag-minImag)+minImag);
rxSym(:,snrIdx) = r;



exactLLR(:,snrIdx) = qamdemod(r,M,symOrder,...
'UnitAveragePower',1,'OutputType','llr','NoiseVariance',noiseVariance);
approxLLR(:,snrIdx) = qamdemod(r,M,symOrder,...
'UnitAveragePower',1,'OutputType','approxllr','NoiseVariance',noiseVariance);
end


nnInput = zeros(numSymbols,2,numSNRValues);
nnOutput = zeros(numSymbols,k,numSNRValues);
for snrIdx = 1:numSNRValues
rxTemp = rxSym(:,snrIdx);
rxTemp = [real(rxTemp) imag(rxTemp)];
nnInput(:,:,snrIdx) = rxTemp;


llrTemp = exactLLR(:,snrIdx);
nnOutput(:,:,snrIdx) = reshape(llrTemp, k, numSymbols)';
end


hiddenLayerSize = 8; trainedNetworks = cell(1,numSNRValues);
for snrIdx=1:numSNRValues
fprintf('Training neural network for SNR = %1.1fdB', ...
SNRValues(snrIdx))
x = nnInput(:,:,snrIdx)';
y = nnOutput(:,:,snrIdx)';


MSExactLLR = mean(y(:).pow(2);
fprintf('Mean Square LLR = %1.2f', MSExactLLR)


mse = inf;
for p=1:5 %Modificación hecha para aumentar el número de pruebas
```

```
netTemp = llrnetNeuralNetwork(hiddenLayerSize);
if parallelComputingLicenseExists()
netTemp,tr
= train(netTemp,x,y,'useParallel','yes');
else
netTemp,tr
= train(netTemp,x,y);
end
% Test the Network
predictedLLRSNR = netTemp(x);
mseTemp = perform(netTemp,y,predictedLLRSNR);
fprintf('tTrial %d: MSE = %1.2e', p, mseTemp)
if mse >mseTempv mse = mseTemp;
net = netTemp;
end
end

% Store the trained network
trainedNetworkssnrIdx = net;
fprintf('Best MSE = %1.2e', mse)
end


numBits = numSymbols*k;
d = randi([0 1], numBits, 1);


txSym = qammod(d,M,symOrder,'InputType','bit','UnitAveragePower',1);


exactLLR = zeros(numBits,numSNRValues);
approxLLR = zeros(numBits,numSNRValues);
predictedLLR = zeros(numBits,numSNRValues);
rxSym = zeros(length(txSym),numSNRValues);
for snrIdx = 1:numSNRValues
SNR = SNRValues(snrIdx);
sigmas = 10pow(-SNR/10);
r = awgn(txSym,SNR);
rxSym(:,snrIdx) = r;
```

```
exactLLR(:,snrIdx) = qamdemod(r,M,symOrder,...
'UnitAveragePower',1,'OutputType','llr','NoiseVariance',sigmas);
approxLLR(:,snrIdx) = qamdemod(r,M,symOrder,...
'UnitAveragePower',1,'OutputType','approxllr','NoiseVariance',sigmas);

net = trainedNetworkssnrIdx;
x = [real(r) imag(r)]';
tempLLR = net(x);
predictedLLR(:,snrIdx) = reshape(tempLLR, numBits, 1);
end

qam16Results.exactLLR = exactLLR;
qam16Results.approxLLR = approxLLR;
qam16Results.predictedLLR = predictedLLR;
qam16Results.RxSymbols = rxSym;
qam16Results.M = M;
qam16Results.SNRValues = SNRValues;
qam16Results.HiddenLayerSize = hiddenLayerSize;
qam16Results.NumSymbols = numSymbols;

trainNow = false;
if trainNow
% Parameters for 64-QAM
simParams(1).M = 64;
simParams(1).SNRValues = 0:5:10;
simParams(1).HiddenLayerSize = 16;
simParams(1).NumSymbols = 1e4;
simParams(1).UseReLU = false;

% Parameters for 256-QAM
simParams(2).M = 256;
simParams(2).SNRValues = 0:10:20;
simParams(2).HiddenLayerSize = 32;
simParams(2).NumSymbols = 1e4;
```

```
simParams(2).UseReLU = false;


simResults = llrnetQAMLLR(simParams);
llrnetPlotLLR(simResults(1),sprintf('%d-QAM LLR Comparison',simResults(1).M))
llrnetPlotLLR(simResults(2),sprintf('%d-QAM LLR Comparison',simResults(2).M))
else
load('llrnetQAMPerformanceComparison.mat', 'simResults')
for p=1:length(simResults)
llrnetPlotLLR(simResults(p),sprintf('%d-QAM LLR Comparison',simResults(p).M))
end
end


————————- PARTE 2————————-
simulateNow = false;
if simulateNow
subsystemType = '16APSK 2/3';
EsNoValues = 8.6:0.1:8.9;
numFrames = 10000;
numErrors = 200;


trainNow = false;
if trainNow  (strcmp(subsystemType,'16APSK 2/3')
isequal(EsNoValues,8.6:0.1:9))
% Train the networks for each EsNo value
numTrainSymbols = 1e4;
hiddenLayerSize = 64;
llrNets = llrnetTrainDVBS2LLRNetwork(subsystemType, EsNoValues, numTrainSymbols, hid-
denLayerSize);
else
load('llrnetDVBS2Networks','llrNets','subsystemType','EsNoValues');
end


% Simulate PER with exact LLR, approximate LLR, and LLRNet [perLLR,perApproxLLR,perLLRNet]
= llrnetDVBS2PER(subsystemType,EsNoValues,llrNets,numFrames,numErrors);
llrnetPlotLLRvsEsNo(perLLR,perApproxLLR,perLLRNet,EsNoValues,subsystemType)
```

```
else
load('llrnetDVBS2PERResults.mat','perApproxLLR','perLLR','perLLRNet',...
'subsystemType','EsNoValues');
llrnetPlotLLRvsEsNo(perLLR,perApproxLLR,perLLRNet,EsNoValues,subsystemType)
end
```

Universidad de Alcalá
Escuela Politécnica Superior

Universidad
de Alcalá