



Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería en Telecomunicaciones

**ANÁLISIS DE LA APLICACIÓN DE AUTO CODIFICADORES EN SISTEMAS DE
COMUNICACIONES PARA MINIMIZAR EL EFECTO DEL RUIDO**

Trabajo de titulación previo a la
obtención del título de Ingeniero
en Telecomunicaciones

Autores:

David Andrés Otavalo Alvarado

Samantha Fernanda Zaruma Morocho

Director:

Lizandro Damián Solano Quinde

ORCID: 0000-0001-7427-4889

Cuenca, Ecuador

2024-06-24

Resumen

Los sistemas de comunicación convencionales utilizan bloques de procesamiento para realizar tareas específicas, como la codificación o modulación de las señales de entrada. Recientemente, la teoría del aprendizaje profundo (DL) ha inspirado a programadores e investigadores en el área de comunicaciones digitales a aplicarla en estos sistemas tradicionales con el objetivo de hacerlos más óptimos y adaptables a diferentes entornos, reduciendo la complejidad en el diseño de bloques que ejecutan tareas específicas. En este contexto, el presente trabajo tiene como objetivo principal desarrollar un auto codificador, que es la aplicación de una red neuronal artificial de aprendizaje no supervisado, con una estructura de codificación-decodificación similar a la de un sistema de comunicación convencional.

El modelo que se presenta en los Capítulos 4 y 5 se desarrolla utilizando las librerías y API funcional de TensorFlow. La codificación implementada se hizo en función de los siguientes esquemas de modulación: BPSK, QPSK, 8-PSK, 16-PSK, 32-PSK, 4-QAM, 16-QAM, 32-QAM, 64-QAM, and 128-QAM. Con el entrenamiento, el modelo busca reducir el error entre la información original y la reconstruida tras pasar por un canal ruidoso AWGN de manera eficiente. Para optimizar el entrenamiento se incorpora la técnica de parada temprana, la cual detiene el entrenamiento cuando los valores de la función de pérdida dejan de mejorar.

Para verificar el funcionamiento del auto codificador (AE), se obtuvieron las gráficas del diagrama de constelación, las curvas de Tasa de error de Bloque (BLER) y Tasa de error de Bit (BER) para cada esquema de modulación. Estas gráficas fueron analizadas y comparadas con las curvas típicas de las modulaciones digitales mencionadas anteriormente. Los resultados del auto codificador reflejan una mayor precisión y eficiencia en la reconstrucción de los datos transmitidos y en consecuencia una mejora significativa en la transmisión de información.

Palabras clave: Auto codificador (AE), AWGN, BLER, BER, parada temprana, entrenamiento, codificación, , activación, constelación, pérdida, tasa, bits, redundancia.

Abstract

Conventional communication systems use processing blocks to perform specific tasks, such as encoding or modulating input signals. Recently, Deep Learning (DL) theory has inspired programmers and researchers in the field of digital communications to apply it to these traditional systems with the aim of making them more optimal and adaptable to different environments, thereby reducing the complexity of having separate blocks for each specific task. In this context, the main objective of this work is to develop an autoencoder, which is essentially an unsupervised learning neural network with an encoding-decoding structure similar to that of a conventional communication system.

The model presented in Chapters 4 and 5 is developed using the TensorFlow libraries and functional API. The implemented encoding was based on the following modulation schemes: BPSK, QPSK, 8-PSK, 16-PSK, 32-PSK, 4-QAM, 16-QAM, 32-QAM, 64-QAM, and 128-QAM. Through training, the model aims to efficiently reduce the error between the original information and the reconstructed information after passing through a noisy AWGN channel. To optimize the training, the early stopping technique is incorporated, which halts the training when the loss function values stop improving.

To verify the functionality of the Autoencoder (AE), constellation diagrams, Block Error Rate (BLER) and Bit Error Rate (BER) plots were obtained for each modulation scheme. These graphs were analyzed and compared with the typical curves of the aforementioned digital modulations. The results of the autoencoder indicate greater precision and efficiency in reconstructing the transmitted data, consequently leading to a significant improvement in information transmission.

Keywords: Autoencoder (AE), AWGN, BLER, BER, early stopping, training, coding, activation, constellation, loss, rate, bits, redundancy.

Índice

Resumen	2
Abstract	3
Índice	4
Índice de figuras	9
Índice de tablas	14
Agradecimientos	16
Dedicatoria	18
1. Introducción	23
1.1. Identificación del Problema	23
1.2. Estado del Arte	24
1.3. Justificación	26
1.4. Alcance	26
1.5. Objetivos	27
1.5.1. Objetivo general	27
1.5.2. Objetivos específicos	27
2. Marco Teórico	28
2.1. Generalidades de los Sistemas de Comunicación	28
2.1.1. Elementos básicos de un sistema de comunicación	28
2.1.2. Ruido e interferencias	29
2.1.2.1. Ruido Blanco Aditivo Gausiano	29
2.1.2.2. Relación señal a ruido	29
2.1.2.3. Tasa de bit erróneo	30
2.1.2.4. Tasa de error de bloque	30
2.2. Inteligencia Artificial	31
2.2.1. Machine Learning	31
2.2.2. Modalidades del Aprendizaje Automático	31
2.2.2.1. Aprendizaje Automático Supervisado	31
2.2.2.2. Aprendizaje Automático no Supervisado	32

2.2.2.3. Aprendizaje Automático por Refuerzo	32
2.2.3. Deep Learning	32
2.3. Redes de Neuronas Artificiales	33
2.3.1. Perceptrón	33
2.3.2. Redes Monocapa feedforward	35
2.3.3. Redes Multicapa feedforward	35
2.3.4. Redes Neuronales Recurrentes	35
2.3.5. Auto codificador	35
2.3.5.1. Auto codificador variacional	36
2.3.5.2. Auto codificador de atenuación de ruido	37
2.3.5.3. Auto codificador poco denso	37
2.4. Arquitectura de las Redes de Neuronas Artificiales	38
2.4.1. Funciones de Activación	38
2.4.1.1. Sigmoidal	38
2.4.1.2. Tangente hiperbólica	38
2.4.1.3. Rectified Linear Unit	38
2.4.1.4. SoftMax	39
2.4.2. Algoritmos de optimización	39
2.4.2.1. Descenso de gradiente Estocástico	40
2.4.2.2. Descenso de gradiente Estocástico más momentum	41
2.4.2.3. Gradiente acelerado Nesterov	42
2.4.2.4. Adagrad	42
2.4.2.5. Adadelta	43
2.4.2.6. RMSProp	43
2.4.2.7. Adam	43
2.4.3. Funciones de pérdida	43
2.4.3.1. Error Cuadrático Medio (MSE)	44
2.4.3.2. Error Absoluto Medio (MAE)	44
2.4.3.3. Entropía Cruzada Binaria	44
2.4.3.4. Entropía Cruzada Categórica	45
2.4.4. Técnicas de Regularización	45
2.4.4.1. Regularización de peso L1	45
2.4.4.2. Regularización de peso L2	46
2.4.4.3. Dropout	46

2.4.4.4. Aumento de Datos (Data Augmentation)	46
2.4.4.5. Parada Temprana (Early Stopping)	46
2.4.4.6. Normalización por lotes (Batch Normalization)	47
2.5. Reducción de Dimensionalidad	48
2.5.1. Análisis de Componentes Principales (PCA)	49
2.5.2. t-Distributed Stochastic Neighbor Embedding (t-SNE)	49
2.5.3. Uniform Manifold Approximation and Projection (UMAP)	50
3. Trabajos Relacionados	52
3.1. Implementación de auto codificadores extremo a extremo	52
3.2. Limitaciones en la implementación de auto codificadores	54
3.2.1. Tendencia a cero o infinito del gradiente	54
3.2.2. Sobreajuste	54
3.2.3. Conclusiones	55
4. Diseño e Implementación el auto codificador	57
4.1. Análisis previo para la arquitectura	57
4.2. Arquitectura para el Codificador	58
4.2.1. Datos de Entrada	58
4.2.2. Capa de entrada	60
4.2.3. Capa Densa	60
4.2.4. Capa de Normalización	61
4.2.4.1. Energía	61
4.2.4.2. Potencia promedio	62
4.3. Arquitectura del Canal	64
4.4. Arquitectura del Decodificador	66
4.4.1. Capa Densa 1	67
4.4.2. Capa Densa 2	67
4.4.3. Datos de Salida	68
4.5. Entrenamiento de la Red	69
4.5.1. Generar modelo	69
5. Análisis y Resultados	72
5.1. Métricas de evaluación para el desempeño del auto codificador	72
5.2. BPSK - auto codificador (2,1)	73
5.2.1. Entrenamiento	73

5.2.2. Diagrama de Constelación	75
5.2.3. Probabilidad de Error	76
5.3. QPSK - auto codificador (n,2)	79
5.3.1. Entrenamiento	79
5.3.2. Diagrama de Constelación	81
5.3.3. Probabilidad de Error	83
5.4. 8-PSK - auto codificador (n,3)	87
5.4.1. Entrenamiento	87
5.4.2. Diagrama de Constelación	90
5.4.3. Probabilidad de Error	92
5.5. 16-PSK - auto codificador (n,4)	94
5.5.1. Entrenamiento	94
5.5.2. Diagrama de Constelación	96
5.5.3. Probabilidad de Error	98
5.6. 32-PSK - auto codificador (n,5)	101
5.6.1. Entrenamiento	101
5.6.2. Diagrama de Constelación	103
5.6.3. Probabilidad de Error	105
5.7. 4-QAM - auto codificador (n,2)	108
5.7.1. Entrenamiento	108
5.7.2. Diagrama de Constelación	111
5.7.3. Probabilidad de Error	112
5.8. 16-QAM - auto codificador (n,4)	115
5.8.1. Entrenamiento	115
5.8.2. Diagrama de Constelación	118
5.8.3. Probabilidad de Error	119
5.9. 32-QAM - auto codificador (n,5)	122
5.9.1. Entrenamiento	122
5.9.2. Diagrama de Constelación	125
5.9.3. Probabilidad de Error	127
5.10.64-QAM - auto codificador (n,6)	130
5.10.1. Entrenamiento	130
5.10.2. Diagrama de Constelación	133
5.10.3. Probabilidad de Error	136

5.11. 128-QAM - auto codificador (n,7)	138
5.11.1. Entrenamiento	138
5.11.2. Diagrama de Constelación	140
5.11.3. Probabilidad de Error	142
5.12. Resumen de resultados y Discusión	145
5.12.1. Análisis de los Resultados de Entrenamiento	145
5.12.2. Análisis de los Resultados de los Diagramas de Constelación	147
5.12.3. Análisis de los Resultados de BLER	148
5.12.4. Análisis de los Resultados de BER	149
5.12.5. Análisis sobre requerimientos de Ancho de Banda	149
6. Conclusiones y recomendaciones	153
6.1. Conclusiones	153
6.2. Recomendaciones	154
6.3. Trabajos Futuros	155
A. Apéndice A	156
A.1. Ejemplo de codificación <i>One Hot</i>	156
B. Apéndice B	157
B.1. Valores para el diagrama de constelación de auto codificadores (2,k) normalizados en energía.	157
B.2. Valores para el diagrama de constelación de auto codificadores (2,k) normalizados en potencia.	158
C. Apéndice C	162
C.1. Valores de BLER para auto codificadores (n,k) normalizados en energía.	162
C.2. Valores de BLER para auto codificadores (n,k) normalizados en potencia.	163
D. Apéndice D	166
D.1. Valores de BER para auto codificadores (n,k) normalizados en energía.	166
D.2. Valores de BER para auto codificadores (n,k) normalizados en potencia.	167
Referencias	170

Índice de figuras

2.1. Diagrama básico de bloques de un sistema de comunicaciones. Fuente:autores	28
2.2. Distintos tipos de aprendizaje en función de los datos. Fuente: [1]	32
2.3. Red Neuronal Profunda con Estructura básica. Fuente: [2]	33
2.4. Estructura del Perceptrón. Fuente: [3]	34
2.5. Tipos de auto codificadores. Fuente: [4]	36
2.6. Funcionamiento del Descenso del Gradiente Estocástico. Fuente: [5]	40
3.1. Un sistema de comunicaciones representado como un AE. Fuente: [6].	52
4.1. Propuesta de Esquema de auto codificador. Fuente: Autores.	58
5.1. Resultados del entrenamiento para el auto codificador (2,1) normalizado en energía.	74
5.2. Diagrama de constelación para el auto codificador (2,1) normalizado en energía.	76
5.3. Gráfica del BLER para auto codificador (2,1) normalizado en energía.	77
5.4. Gráfica del BER para el auto codificador (2,1) normalizado en energía.	79
5.5. Resultados del entrenamiento para el auto codificador (2,2) normalizado en energía.	80
5.6. Resultados del entrenamiento para el auto codificador (3,2) normalizado en energía.	80
5.7. Resultados del entrenamiento para el auto codificador(4,2) normalizado en energía.	81
5.8. Diagrama de constelación para el auto codificador (2,2) normalizado en energía.	82
5.9. Constelaciones proyectadas para el auto codificador (3,2) normalizado en energía.	83
5.10. Constelaciones proyectadas para el auto codificador (4,2) normalizado en energía.	83
5.11. Gráfica del BLER para auto codificador (n,2) normalizado en energía.	85
5.12. Gráfica del BLER para el auto codificador (n,2) normalizado en energía.	86
5.13. Resultados del entrenamiento para el auto codificador (2,3) normalizado en energía.	87
5.14. Resultados del entrenamiento para el auto codificador (3,3) normalizado en energía.	88
5.15. Resultados del entrenamiento para el auto codificador (4,3) normalizado en energía.	89

5.16. Resultados del entrenamiento para el auto codificador (6,3) normalizado en energía.	89
5.17. Diagrama de constelación para el auto codificador (2,3) normalizado en energía.	90
5.18. Constelaciones proyectadas para el auto codificador (3,3) normalizado en energía.	91
5.19. Constelaciones proyectadas para el auto codificador (4,3) normalizado en energía.	91
5.20. Constelaciones proyectadas para el auto codificador (6,3) normalizado en energía.	91
5.21. Gráfica del BLER para el auto codificador (n,3) normalizado en energía.	92
5.22. Gráfica del BER para el auto codificador (n,3) normalizado en energía.	93
5.23. Resultados del entrenamiento para el auto codificador (2,4) normalizado en energía.	94
5.24. Resultados del entrenamiento para el auto codificador (4,4) normalizado en energía.	95
5.25. Resultados del entrenamiento para el auto codificador (6,4) normalizado en energía.	95
5.26. Resultados del entrenamiento para el auto codificador (8,4) normalizado en energía.	96
5.27. Diagrama de constelación para el auto codificador (2,4) normalizado en energía.	97
5.28. Constelaciones proyectadas para el auto codificador (4,4) normalizado en energía.	97
5.29. Constelaciones proyectadas para el auto codificador (6,4) normalizado en energía.	98
5.30. Constelaciones proyectadas para el auto codificador (8,4) normalizado en energía.	98
5.31. Gráfica del BLER para el auto codificador (n,4) normalizado en energía.	99
5.32. Gráfica del BER para el auto codificador (n,4) normalizado en energía.	100
5.33. Resultados del entrenamiento para el auto codificador (2,5) normalizado en energía.	101
5.34. Resultados del entrenamiento para el auto codificador (5,5) normalizado en energía.	102
5.35. Resultados del entrenamiento para el auto codificador (6,5) normalizado en energía.	103
5.36. Resultados del entrenamiento para el auto codificador (10,5) normalizado en energía.	103
5.37. Diagrama de constelación para el auto codificador (2,5) normalizado en energía.	104
5.38. Constelaciones proyectadas para el auto codificador (5,5) normalizado en energía.	105
5.39. Constelaciones proyectadas para el auto codificador (6,5) normalizado en energía.	105

5.40. Constelaciones proyectadas para el auto codificador (10,5) normalizado en energía.	106
5.41. Gráfica del BLER para el auto codificador (n,5) normalizado en energía.	107
5.42. Gráfica del BER para el auto codificador (n,5) normalizado en energía.	108
5.43. Resultados del entrenamiento para el auto codificador (2,2) normalizado en potencia.	109
5.44. Resultados del entrenamiento para el auto codificador (3,2) normalizado en potencia.	110
5.45. Resultados del entrenamiento para el auto codificador (4,2) normalizado en potencia.	110
5.46. Diagrama de constelación para el auto codificador (2,2) normalizado en potencia.	111
5.47. Constelaciones proyectadas para el auto codificador (3,2) normalizado en potencia.	112
5.48. Constelaciones proyectadas para el auto codificador (4,2) normalizado en potencia.	113
5.49. Gráfica del BLER para el auto codificador (n,2) normalizado en potencia.	114
5.50. Gráfica del BER para auto codificador (n,2) normalizado en potencia.	114
5.51. Resultados del entrenamiento para el auto codificador (2,4) normalizado en potencia.	115
5.52. Resultados del entrenamiento para el auto codificador (4,4) normalizado en potencia.	116
5.53. Resultados del entrenamiento para el auto codificador (6,4) normalizado en potencia.	117
5.54. Resultados del entrenamiento para el auto codificador (8,4) normalizado en potencia.	117
5.55. Diagrama de constelación para el auto codificador (2,4) normalizado en potencia.	118
5.56. Constelaciones proyectadas para el auto codificador (4,4) normalizado en potencia.	119
5.57. Constelaciones proyectadas para el auto codificador (6,4) normalizado en potencia.	120
5.58. Constelaciones proyectadas para el auto codificador (8,4) normalizado en potencia.	120
5.59. Gráfica del BLER para auto codificador (n,4) normalizado en potencia.	121
5.60. Gráfica del BER para auto codificador (n,4) normalizado en potencia.	122

5.61.Resultados del entrenamiento para el auto codificador (2,5) normalizado en potencia.	123
5.62.Resultados del entrenamiento para el auto codificador (5,5) normalizado en potencia.	124
5.63.Resultados del entrenamiento para el auto codificador (6,5) normalizado en potencia.	124
5.64.Resultados del entrenamiento para el auto codificador (10,5) normalizado en potencia.	125
5.65.Diagrama de constelación para el auto codificador (2,5) normalizado en potencia.	126
5.66.Constelaciones proyectadas para el auto codificador (5,5) normalizado en potencia.	127
5.67.Constelaciones proyectadas para el auto codificador (6,5) normalizado en potencia.	127
5.68.Constelaciones proyectadas para el auto codificador (10,5) normalizado en potencia.	128
5.69.Gráfica del BLER para auto codificador (n,5) normalizado en potencia.	129
5.70.Gráfica del BER para auto codificador (n,5) normalizado en potencia.	130
5.71.Resultados del entrenamiento para el auto codificador (2,6) normalizado en potencia.	131
5.72.Resultados del entrenamiento para el auto codificador (6,6) normalizado en potencia.	132
5.73.Resultados del entrenamiento para el auto codificador (9,6) normalizado en potencia.	132
5.74.Resultados del entrenamiento para el auto codificador (12,6) normalizado en potencia.	133
5.75.Diagrama de constelación para el auto codificador (2,6) normalizado en potencia.	134
5.76.Constelaciones proyectadas para el auto codificador (6,6) normalizado en potencia.	135
5.77.Constelaciones proyectadas para el auto codificador (9,6) normalizado en potencia.	135
5.78.Constelaciones proyectadas para el auto codificador (12,6) normalizado en potencia.	135
5.79.Gráfica del BLER para auto codificador (n,6) normalizado en potencia.	137
5.80.Gráfica del BER para auto codificador (n,6) normalizado en potencia.	138

5.81.Resultados del entrenamiento para el auto codificador (2,7) normalizado en po-	
tencia.	139
5.82.Resultados del entrenamiento para el auto codificador (7,7) normalizado en po-	
tencia.	139
5.83.Resultados del entrenamiento para el auto codificador (14,7) normalizado en	
potencia.	140
5.84.Diagrama de constelación para el auto codificador (2,7) normalizado en potencia.	141
5.85.Constelaciones proyectadas para el auto codificador (7,7) normalizado en po-	
tencia.	142
5.86.Constelaciones proyectadas para el auto codificador (14,7) normalizado en po-	
tencia.	142
5.87.Gráfica del BLER para auto codificador (n,7) normalizado en potencia.	143
5.88.Gráfica del BER para auto codificador (n,7) normalizado en potencia.	144

Índice de tablas

4.1. Ejemplo de codificación <i>One Hot</i> para Quadrature Phase Shift Keying (QPSK)	59
4.2. Ejemplo de codificación generalizada <i>One Hot</i>	59
4.3. Parámetros de Entrenamiento	70
5.1. Valores del auto codificador (2,1) normalizado en energía.	75
5.2. Valores de BLER con su respectivo IC para el auto codificador (2,1) normalizado en energía.	77
5.3. Valores de BER con su respectivo IC para el auto codificador (2,1) normalizado en energía.	78
5.4. Valores del auto codificador (2,2) normalizado en energía.	82
5.5. Valores del auto codificador (4,2) normalizado en energía.	82
5.6. Valores de BLER con su respectivo IC para el auto codificador (n,2) normalizado en energía.	84
5.7. Valores de BER con su respectivo IC para el auto codificador (n,2) normalizado en energía.	86
5.8. Resultados finales del entrenamiento para los diferentes auto codificadores normalizados en energía.	146
5.9. Resultados finales del entrenamiento para los diferentes auto codificadores normalizados en potencia.	147
5.10. Ancho de banda eficiente para M-ary Phase Shift Keying (M-PSK).	150
5.11. Ancho de banda eficiente para M-ary Quadrature Amplitude Modulation (M-QAM).	150
5.12. Factor de expansión en el Ancho de Banda al emplear codificación de canal.	151
A.1. Ejemplo de codificación <i>One Hot</i> para 16-QAM	156
B.1. Valores del auto codificador (2,3) normalizado en energía.	157
B.2. Valores del auto codificador (2,4) normalizado en energía.	157
B.3. Valores del auto codificador (2,5) normalizado en energía.	157
B.4. Valores del auto codificador (2,2) normalizado en potencia.	158
B.5. Valores del auto codificador (2,4) normalizado en potencia.	158
B.6. Valores del auto codificador (2,5) normalizado en potencia.	158
B.7. Valores del auto codificador (2,6) normalizado en potencia.	159
B.8. Valores del auto codificador (2,7) normalizado en potencia - Parte 1.	160
B.9. Valores del auto codificador (2,7) normalizado en potencia - Parte 2.	161

C.1. Valores de BLER con su respectivo IC para el auto codificador (n,3) normalizado en energía.	162
C.2. Valores de BLER con su respectivo IC para el auto codificador (n,4) normalizado en energía.	162
C.3. Valores de BLER con su respectivo IC para el auto codificador (n,5) normalizado en energía.	163
C.4. Valores de BLER con su respectivo IC para el auto codificador (n,2) normalizado en potencia.	163
C.5. Valores de BLER con su respectivo IC para el auto codificador (n,4) normalizado en potencia.	164
C.6. Valores de BLER con su respectivo Intervalo de Confianza (IC) para el auto codificador (n,5) normalizado en potencia.	164
C.7. Valores de BLER con su respectivo IC para el auto codificador (n,6) normalizado en potencia.	165
C.8. Valores de BLER con su respectivo IC para el auto codificador (n,7) normalizado en potencia.	165
D.1. Valores de BER para el auto codificador (n,3) normalizado en energía.	166
D.2. Valores de BER para el auto codificador (n,4) normalizado en energía.	166
D.3. Valores de BER para el auto codificador (n,5) normalizado en energía.	167
D.4. Valores de BER con su respectivo IC para el auto codificador (n,2) normalizado en potencia.	167
D.5. Valores de BER con su respectivo IC para el auto codificador (n,4) normalizado en potencia.	168
D.6. Valores de BER con su respectivo IC para el auto codificador (n,5) normalizado en potencia.	168
D.7. Valores de BER con su respectivo IC para el auto codificador (n,6) normalizado en potencia.	169
D.8. Valores de BER con su respectivo IC para el auto codificador (n,7) normalizado en potencia.	169

Agradecimientos.

A **Dios**, por brindarme la salud para alcanzar y cumplir las metas que me he propuesto.

A mis padres, **Marta y José**, quienes siempre estuvieron presentes con su apoyo, sus consejos y, sobre todo, alentándome a cumplir esta meta. Nunca se cansaron ni dejaron de creer en mí.

A **Daniela**, quien desde el primer día fue la compañera perfecta para compartir los momentos buenos y malos que enfrenté durante toda esta etapa universitaria. Gracias por el apoyo y por darme las fuerzas para levantarme de los tropiezos y seguir adelante hasta alcanzar la meta.

A mi hermana **Cristina**, por ser un ejemplo y, sobre todo, por enseñarme que siempre se puede continuar. A mi prima **Guadalupe**, primos, primas y demás familiares, por cada encuentro que se convirtió en un momento para compartir palabras de motivación.

A mis amigos **Carlos, Diego, Katherine, Andrés, July, Alexandra, Franklin, Fabricio** y los miembros de Machetes, con quienes compartí buenos momentos. Gracias por demostrar que no todo era estudio, sino que también había tiempo para reír, cantar, bailar, jugar, discutir o simplemente celebrar un cumpleaños.

A mi amiga, compañera de tesis y futura colega, **Samantha**, por todo el esfuerzo y sacrificio realizado para cumplir con este objetivo. Sin duda, que luego de una tercera oportunidad, hoy se están dando los frutos. Gracias, por el tiempo, la compañía y sobre todos las enseñanzas que compartimos. Te deseo éxitos en tu nueva etapa profesional y recuerda siempre confiar en ti.

Al tutor, **Ing. Lizandro Solano** por su tiempo y orientación en cada reunión, su aporte fue una contribución valiosa para solventar dudas y cumplir con éxito el trabajo de titulación. Al **Ing. Darwin Astudillo** por el apoyo y la predisposición a colaborar en cualquier momento. Finalmente, a la universidad que se convirtió en una segunda casa durante todo esta etapa. Con orgullo 100 % Ucuencia.

D. Andrés Otavalo Alvarado

A **Dios** y a la virgen **Guardiana de la Fe** que con su infinito amor me brindaron salud, fortaleza y sobre todo sabiduría para culminar este largo trayecto universitario.

A mi madre y mejor amiga, **Fernanda**, quien con su amor incondicional ha estado presente en todas las etapas de mi vida siendo la estrella que me guía por el mejor camino. Gracias por tus consejos, tu apoyo y comprensión en este largo camino; pero sobre todo por ser ese ejemplo de valentía ante las dificultades de la vida. ¡Tú guerrera ha terminado una batalla!

A mi padre, **Mauricio**, que al ser mi ejemplo de vida más grande me ayudó a no darme por vencida por más difícil que fuera la situación. Gracias por los consejos motivadores en el momento justo, por tu ejemplo de superación y el amor infinito que demuestras a tus hijos.

A mis hermanos, **Mateo y Nicolás**, su presencia en mi vida me inspiran a ser mejor persona. Gracias por estar ahí en todo momento, y sentirse orgullosos de mí, esta victoria va para ustedes.

A mi cómplice de vida, **Chuz**, su compañía y apoyo incondicional ha hecho que cada desafío valga la pena en toda esta etapa universitaria. Gracias por inspirarme y formar parte de este gran sueño. ¡El logro es de los dos, juntos hasta viejitos!

A mi compañero de tesis, mejor amigo y futuro colega, **David**, por su esfuerzo, sacrificio y dedicación en el desarrollo de este trabajo . Gracias por los consejos, el apoyo, la paciencia, el tiempo y las experiencias vividas, haz sido un pilar fundamental en la recta final.

A mis hermanos, **Gladiadores**, que sin ellos la vida universitaria habría sido distinta. Gracias por enseñarme la hermandad y el camino de Dios.

A mis amigos, **Machetes V3.0**, por enseñarme el significado de la amistad y el trabajo en equipo. Lo logramos amigos, hemos cumplido el gran sueño.

Al tutor, **Ing. Lizandro Solano**, por depositar su confianza en cada uno de nosotros, además del tiempo y su apoyo incondicional para la culminación de este proyecto. Su gran personalidad, profesionalismo y ética son cualidades que resalto y admiro.

Al, **Ing. Darwin Astudillo**, por el apoyo, la disposición y el espíritu de colaboración brindados para aclarar dudas en cualquier momento durante estos dos últimos semestres. A la, **Ing. Priscila Cedillo**, por su apoyo y sobre todo confianza durante esta etapa universitaria. Su gran calidad humana y profesional me inspiraron para cumplir mi objetivo.

Samantha F. Zaruma Morocho

Dedicatoria.

Dedicado a mis padres, quienes estarán orgullosos y celebrarán este logro como propio, sabiendo que su esfuerzo ha dado sus frutos. A mi abuelo y abuela, tía y tío, quienes ya no me acompañan, pero cuyos consejos y ejemplos de vida fueron mi motivación para superarme día a día. A mi abuela materna por cuidarme a la distancia.

Dedicado a Daniela, quien es la única que conoce de principio a fin todo el esfuerzo realizado y ha comprendido muchas veces la necesidad de no compartir ciertos momentos. Este logro también es tuyo, y deseo verte feliz y orgullosa cada día.

Dedicado a mi sobrino Nicolás, mis sobrinas Micaela y Danae, mi ahijada María Belén, quienes desde su llegada han sido una gran motivación. Que Dios les brinde la salud para crecer y construir su propio camino, hasta convertirse en los próximos profesionales de la familia.

D. Andrés Otavallo Alvarado

Con inmenso cariño y amor, deseo dedicar esta tesis a mis padres, que sin importar las dificultades que se han presentado a lo largo de esta etapa jamás dejaron de apoyarme y sentirse orgullosos de cada paso que daba. El sueño anhelado se cumplió, esto va para ustedes.

A mi angelito del cielo, Sarita, gracias a ella obtuve la fuerza y la bendición para cumplir el sueño anhelado.

A mis hermanos, Mateo y Nicolás que desde el primer día han sido mi fuerza e inspiración para salir adelante y no rendirme jamás. Que Dios me permita ser una guía en su camino y así como logré cumplir mi sueño, pueda ser un apoyo para que ustedes cumplan todo lo que se propongan.

A mi Chuz, que es la persona que conoce de principio a fin todo el esfuerzo, tropiezos, decepciones, logros vividos en esta gran etapa de mi vida. Por siempre ser ese hombro que me brinda apoyo y tener siempre un consejo para impulsarme a continuar. Tu amor es una herramienta fundamental para cumplir mis sueños.

A mis abuelitos, tíos y primos, por estar conmigo en las buenas y en las malas. Gracias por enseñarme que la familia es el mejor refugio y fuerza para la época de crisis y el mejor lugar para celebrar los logros conseguidos.

A mis perrihijos que estuvieron conmigo en mis desvelos, rabietas y decepciones lo logramos.

¡Somos Ingenieros!

“El tiempo de Dios es perfecto ” Eclesiastés 3,1

Samantha F. Zaruma Morocho

Glosario

128-QAM 128-Quadrature Amplitude Modulation. 2, 3, 8, 138, 140, 142–144, 147

16-PSK 16-Phase Shift Keying. 2, 3, 7, 94, 96, 98–100

16-QAM 16-Quadrature Amplitude Modulation. 2, 3, 7, 14, 59, 115, 119, 121, 122, 147, 148, 156

32-PSK 32-Phase Shift Keying. 2, 3, 7, 101, 103, 105–107

32-QAM 32-Quadrature Amplitude Modulation. 2, 3, 7, 122, 125, 127–129, 147

4-QAM 4-Quadrature Amplitude Modulation. 2, 3, 7, 108, 111, 113, 114, 147

64-QAM 64-Quadrature Amplitude Modulation. 2, 3, 7, 130, 133, 136, 137, 147

8-PSK 8-Phase Shift Keying. 2, 3, 7, 87, 90, 92, 93

Adadelta Adaptive Delta Gradient. 43, 53

Adagrad Adaptive Gradient Algorithm. 42, 43, 53

Adam Adaptive Moment Estimation. 43, 53

Adamax Infinite norm based on Adam. 53

AE Autoencoder. 2, 3, 24, 35–37, 52, 57, 72, 76, 78, 79, 82, 84, 92, 93, 98–100, 103, 105, 106, 112, 113, 115, 120, 122, 128, 129, 136, 137, 142–144, 148, 149, 151, 153–155

ANN Artificial Neural Networks. 31

AWGN Additive White Gaussian Noise. 2, 3, 24, 25, 29, 64, 65, 72

BER Bit Error Rate. 2, 3, 8–15, 26, 27, 30, 54, 72, 77–79, 84–86, 93, 100, 107, 108, 113, 114, 121, 122, 129, 130, 136–138, 144, 149, 150, 154, 166–169

BLER Block Error Rate. 2, 3, 8–15, 24, 25, 30, 53, 72, 76–78, 83–86, 92, 93, 98–100, 105–107, 112–114, 119, 121, 122, 127–130, 136, 137, 142–144, 148, 149, 154, 162–165

BPSK Binary Phase Shift Keying. 2, 3, 6, 73–76, 78, 79

CNN Convolutional Neural Network. 24, 25

CPU Central Processing Unit. 154

DAE Denoising Autoencoder. 37

DL Deep Learning. 2, 3, 23, 31–33

DNN Deep Dense Neural Network. 24

FNN Feedforward Neural Networks. 52, 53

GPU Graphics Processing Unit. 154

IA Inteligencia Artificial. 23, 31

IC Intervalo de Confianza. 14, 15, 77, 78, 84, 86, 92, 99, 106, 113, 120, 128, 136, 143, 162–165, 167–169

IoT Internet Of Things. 154

M-PSK M-ary Phase Shift Keying. 14, 108, 149–151, 153

M-QAM M-ary Quadrature Amplitude Modulation. 14, 150, 151, 153

ML Machine Learning. 23, 31

Nadam Nesterov-accelerated Adaptive Moment Estimation. 42, 53

QPSK Quadrature Phase Shift Keying. 2, 3, 7, 14, 59, 79, 81, 84, 85, 148

ReLU Rectified Linear Unit. 38, 56, 61, 66, 67, 153

RF Rayleigh fading. 25

RMSprop Root Mean Square Propagation. 43, 53

SAE Sparse Autoencoder. 37

SDR Software Defined Radio. 25

SGD Stochastic gradient descent. 40, 41, 53

SNR Signal to noise ratio. 25, 29, 53, 72, 75, 81, 90, 97, 104, 111, 118, 126, 134, 141, 148–150, 154

SVM Support Vector Machine. 24

t-SNE T-distributed Stochastic Neighbor Embedding.. 82, 90, 112, 119, 126, 134, 141

UMAP Uniform Manifold Approximation and Projection. 82, 83, 90, 112, 119, 126, 134, 141

VAE Variational Autoencoder. 36, 37

WGN White Gaussian Noise. 29, 57

1. Introducción

Este capítulo, introduce distintos aspectos que permiten una comprensión completa del contexto y los propósitos del presente trabajo de titulación. La Sección 1.1 aborda la identificación del problema. Como segundo punto, la Sección 1.2 presenta una breve revisión del estado del arte para la implementación del trabajo en cuestión. Posteriormente, la Sección 1.3 plantea la justificación que respalda la relevancia de este trabajo. A continuación, la Sección 1.4 expone el alcance del trabajo. Finalmente, la Sección 1.5 presenta el objetivo general y los objetivos específicos planteados.

1.1. Identificación del Problema

La comunicación confiable entre un origen y un destino a través de un canal involucra la cooperación de un transmisor y un receptor diseñados con múltiples bloques independientes; como codificadores, moduladores y decodificadores [7]. En sistemas tradicionales, se utilizan esquemas de modulación y codificación para transmitir información de manera eficiente y confiable. Sin embargo, la optimización individual de estos bloques no garantiza un rendimiento óptimo global [6]. Por ejemplo, la teoría de la información, establecida por Claude Shannon, establece límites fundamentales en la transmisión de información, como la capacidad del canal y el teorema del canal ruidoso. Sin embargo, no ofrece una guía directa sobre cómo diseñar sistemas de comunicación específicos para alcanzar esos límites. En consecuencia, la optimización práctica implica encontrar estrategias y técnicas que se acerquen a esos límites en condiciones realistas [8].

Las redes neuronales artificiales, el aprendizaje automático (en inglés, Machine Learning (ML)) y el aprendizaje profundo (en inglés, Deep Learning (DL)), son poderosas herramientas de la Inteligencia Artificial (IA), que se han convertido en un área de investigación prometedora en ingeniería de comunicaciones [9]. El uso de la IA permite optimizar sistemas de comunicación de manera global, reduciendo la complejidad en el diseño [9]. Estas técnicas se aplican en diversos aspectos de las comunicaciones, como la estimación del canal, demodulación y detección de modulación [9].

1.2. Estado del Arte

Las técnicas de Machine Learning se pueden dividir en estructuras de una sola capa y de varias capas o multicapa, también conocidas como Deep Learning [10]. En el aprendizaje supervisado, es necesario etiquetar los datos de entrenamiento para que el modelo pueda aprender de ellos. Esto incluye el uso de máquinas de vectores de soporte (Support Vector Machine (SVM)) y redes neuronales convolucionales (Convolutional Neural Network (CNN)), que entran algoritmos para clasificar datos o estimar resultados con precisión [10]. Por otro lado, el aprendizaje no supervisado emplea modelos como máquinas Boltzmann restringidas o auto codificadores, que solo requieren datos de entrenamiento de entrada para inferir patrones de un conjunto de datos sin referencia a resultados conocidos o etiquetados.

Un auto codificador o codificador automático (en inglés, Autoencoder (AE)), es una red neuronal artificial multicapa con aprendizaje no supervisado, que se utiliza para codificar la entrada en una representación comprimida y posteriormente decodificar de manera que la entrada reconstruida sea lo más similar posible a la original [11].

Una primera aplicación de los auto codificadores se encuentra en el procesamiento de imágenes, donde son utilizados para reducir el ruido en radiografías de placas de Rayos X. En este contexto, se alcanza un buen resultado, en términos de robustez frente al ruido, si el auto codificador es capaz de eliminar irregularidades y otras imperfecciones que presentaba la imagen original. Sin embargo, el objetivo de comprimir la imagen no se llega a alcanzar, sin afectar la calidad de las características requeridas, dejando una imagen de salida borrosa y sin contenido [12].

En [13] se presenta un enfoque de aprendizaje no supervisado en un sistema de comunicaciones de extremo a extremo, mediante la optimización de los parámetros para la reconstrucción a través de un auto codificador de canal. Los autores buscan analizar nuevos métodos de modulación que desvanezcan la división entre modulación y corrección de errores; reduciendo a su vez la complejidad computacional y tiempos de ejecución comparados con los requeridos por un sistema de comunicación tradicional. Por lo tanto, los autores comparan dos estructuras de red para un auto codificador, una red neuronal densa y profunda (en inglés, Deep Dense Neural Network (DNN)) y una red neuronal convolucional. Demostraron que para un canal Additive White Gaussian Noise (AWGN) se obtiene el mejor rendimiento utilizando una DNN; sin embargo, los tiempos de entrenamiento son más largos.

En [14] se propone un auto codificador para evaluar la Tasa de Error de Bloque (en inglés, Block

Error Rate (BLER)) y la constelación mediante diversos algoritmos de optimización de extremo a extremo. El codificador automático se entrenó con entropía cruzada categórica logrando un óptimo rendimiento en la Relación Señal a Ruido (en inglés, Signal to noise ratio (SNR)) frente al BLER. No obstante, la complejidad de este codificador automático se centró en el entrenamiento. Para solventar esta situación, en [15] se introduce una nueva arquitectura de auto codificador con menor dimensión en cada capa, con el objetivo de reducir la complejidad en el entrenamiento y procesamiento, obteniendo un rendimiento similar en BLER a sistemas convencionales de codificación y modulación BPSK.

En [16] se presenta una forma alternativa de entrenamiento, basado en retroalimentación, para un codificador automático basado en redes neuronales profundas con múltiples capas densas. Los resultados muestran que los sistemas de comunicación de extremo a extremo entrenados con retroalimentación logran un rendimiento similar en términos de eficiencia espectral al emplear canales ruidosos (AWGN) y desvanecimiento Rayleigh (en inglés, Rayleigh fading (RF)) para un esquema de modulación QPSK.

En [17] se propone un enfoque de aprendizaje profundo para el análisis de diferentes esquemas de modulación digital basado en un codificador automático con eliminación de ruido y una CNN. Mediante GNU radio se generó una señal de bajo ruido para entrenar un modelo de clasificación de esquemas de modulación que puede alcanzar una precisión en la clasificación del 94 %. Sin embargo, el ruido afecta de manera considerable el rendimiento de la red neuronal, por lo que se construyeron varias versiones de codificadores automáticos para eliminación de ruido, con el objetivo de mejorar el rendimiento del modelo de clasificación en señales con SNR bajas.

Finalmente, en [18] se aborda el uso de un auto codificador en el área de las comunicaciones inalámbricas. Los autores implementaron un sistema de comunicación completo basado en redes neuronales que utiliza radio definida por software (en inglés, Software Defined Radio (SDR)), en el que todo un sistema de comunicación se componía únicamente de redes neuronales para su entrenamiento y funcionamiento. Dado que un sistema de este tipo considera la variación de las características del canal en tiempo real, su funcionamiento es comparable al de los sistemas de comunicaciones inalámbricas existentes. Sin embargo, el rendimiento en términos de BLER del prototipo es aproximadamente 1 dB menor que el de un sistema básico DQPSK.

1.3. Justificación

Un enfoque para un sistema de comunicaciones basado en aprendizaje profundo busca optimizar el transmisor y el receptor de extremo a extremo sin depender de estructuras de bloques tradicionales. Esto se logra mediante el uso de auto codificadores que constan de dos partes: el codificador y el decodificador [11]. La función del codificador es comprimir y codificar los datos, mientras que la del decodificador es restaurar los datos convertidos por el codificador a su espacio de representación original. Las capas internas de la arquitectura del codificador automático están diseñadas con parámetros relacionados con el codificador de canal y las funciones del modulador en el transmisor y las funciones del demodulador y decodificador de canal en el receptor.

El algoritmo del codificador automático puede considerarse una gran solución si el error obtenido en la decodificación es el menor posible [11].

1.4. Alcance

La implementación de un auto codificador busca reemplazar la estructura de bloques tradicionales por una estructura optimizada al utilizar el entrenamiento basado en redes neuronales artificiales. Por lo tanto, el trabajo propuesto se centra en la implementación de un auto codificador en una plataforma de programación, con el objetivo de analizar el rendimiento de un sistema de comunicación en comparación con esquemas de modulación digital tradicionales.

La implementación de un auto codificador, basado en una red neuronal no supervisada sigue una metodología estructurada para el aprendizaje de representaciones de datos:

- Seleccionar la estructura de la red neuronal, incluyendo el número de capas ocultas, función de activación, algoritmo de optimización, tasa de aprendizaje, modelo de canal.
- Ajustar iterativamente los pesos de la red en el proceso de entrenamiento para reducir la diferencia entre los datos de entrada y los reconstruidos.
- Representar la constelación de la modulación utilizada dado un conjunto de datos de entrada.

A partir de este trabajo de titulación, se pretende evaluar la simulación del sistema implementado respecto a un sistema de comunicación tradicional. Uno de los criterios más destacados para evaluar el rendimiento de los sistemas es la tasa de error de bit (en inglés, Bit Error Rate

(BER)). En este contexto, la forma de evaluar dicho parámetro en un sistema de comunicación basado en un auto codificador será por medio de la simulación para obtener el BER para cada esquema de modulación, mientras que para el caso de un sistema de comunicación tradicional se utilizarán los valores teóricos.

1.5. Objetivos

1.5.1. Objetivo general

Analizar la aplicación de un auto codificador en un sistema de comunicación para mitigar el efecto del ruido.

1.5.2. Objetivos específicos

- Diseñar y definir la arquitectura, así como los parámetros de activación y optimización, de un codificador automático para minimizar el efecto del canal de un sistema de comunicación.
- Entrenar un auto codificador adaptado a las características del canal de comunicación.
- Evaluar el sistema simulado, con respecto a un sistema de comunicación tradicional.

2. Marco Teórico

Este capítulo introduce los conceptos teóricos fundamentales para el desarrollo del presente trabajo de titulación.

2.1. Generalidades de los Sistemas de Comunicación

2.1.1. Elementos básicos de un sistema de comunicación

Desde un punto de vista muy general, un sistema de comunicación puede definirse como un mecanismo mediante el cual la información producida por una fuente se transmite a un receptor a cierta distancia.

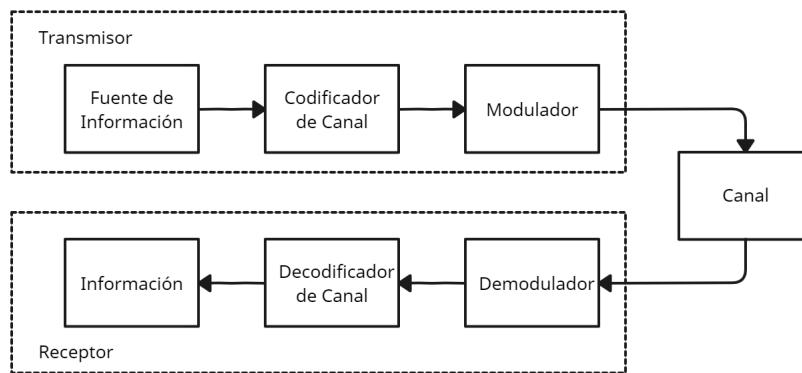


Figura 2.1: Diagrama básico de bloques de un sistema de comunicaciones. Fuente:autores

En este contexto, un sistema de comunicación considera tres elementos básicos: transmisor, canal y receptor. Estos tres componentes se representan mediante un diagrama de bloques, en el esquema de la figura 2.1. El transmisor es responsable de convertir la señal de información a un formato adecuado para su transmisión a través de un medio de comunicación (canal) guiado, como cable coaxial fibra óptica o par trenzado, o no guiado (inalámbrico), como enlaces de radio o microondas [19]. La tarea principal del receptor es procesar la señal recibida de tal manera que pueda reconstruir una forma reconocible de la señal del mensaje original, teniendo en cuenta que está estará atenuada y distorsionada debido a la naturaleza ruidosa del canal [19].

2.1.2. Ruido e interferencias

Se puede considerar como ruido a cualquier señal ajena al receptor que provenga de fuentes que no sean el transmisor. En este sentido, el ruido incluiría las señales de otros usuarios que comparten una parte del espectro o que operan en regiones espectrales cercanas; este tipo de ruido, producido por otros sistemas de comunicaciones, se llama interferencia[19]. Las fuentes de ruido pueden ser internas y/o externas al sistema de comunicación.

2.1.2.1. Ruido Blanco Aditivo Gausiano

El ruido blanco es una señal aleatoria donde los valores en diferentes momentos no están correlacionados, y su densidad espectral de potencia (PSD) es constante. Esto significa que tiene la misma potencia en todas las frecuencias y contiene todas las frecuencias.[20].

El ruido White Gaussian Noise (WGN) presente en el canal, afecta de forma independiente a cada símbolo transmitido. El canal Additive White Gaussian Noise (AWGN) se define como un proceso aleatorio Gaussiano con media cero; es decir, es una función aleatoria caracterizada como una función de densidad de probabilidad Gaussiana [20].

2.1.2.2. Relación señal a ruido

La relación señal a ruido (en inglés, Signal to noise ratio (SNR)) se refiere a la comparación entre el nivel de la señal deseada y el nivel del ruido no deseado, que generalmente incluye interferencias o distorsiones. [20]. La SNR es un indicador de la condición de la señal, un valor alto sugiere una mejor calidad de la información recibida, mientras que un valor más bajo indica que puede ser difícil distinguir la señal deseada del ruido de fondo.

$$\frac{S}{N}(dB) = 10 \log \left(\frac{\text{Potencia Señal}}{\text{Potencia Ruido}} \right) = 10 \log \left(\frac{P_s}{P_N} \right) \quad (2.1)$$

Una relación señal/ruido de baja calidad se define con un mínimo de 10 a 15 dB; en realidad, una relación señal/ruido de 16 a 24 dB ya se considera deficiente. Un valor de entre 25 y 40 dB sería ideal, y más de 41 dB sería las mejores condiciones.

2.1.2.3. Tasa de bit erróneo

La tasa de errores de bits (Bit Error Rate (BER)) mide la integridad de las señales de telecomunicaciones según la cantidad o el porcentaje de bits transmitidos que se reciben incorrectamente. Básicamente, a medida que la calidad de la señal disminuye, aumenta el número de bits incorrectos. El BER es un indicador eficaz del rendimiento de extremo a extremo, ya que abarca tanto el transmisor como el receptor, así como el medio de comunicación entre ellos [20].

La tasa de errores de bits (BER) se calcula dividiendo la cantidad de bits recibidos con errores entre el número total de bits transmitidos en el mismo período de tiempo [20]. Un resultado de 10^{-6} se considera generalmente una tasa de errores de bits aceptable en el ámbito de las telecomunicaciones, mientras que un valor de 10^{-13} es un BER mínimo apropiado para la transmisión de datos. La tasa de errores de bits se puede calcular como:

$$BER = \frac{1}{2} \operatorname{erfc} \left(\frac{E_b}{N_0} \right) = \frac{1}{2} \sqrt{\operatorname{erfc}(SNR)} \quad (2.2)$$

2.1.2.4. Tasa de error de bloque

La Tasa de error de Bloque (Block Error Rate (BLER)) es una métrica de carácter importante utilizada para evaluar la tasa de error en sistemas de comunicación inalámbrica. En estos sistemas, los datos se transmiten en bloques o paquetes, y el BLER indica el porcentaje de estos bloques que contienen errores al ser recibidos por el receptor[21].

Mientras el BER mide la proporción de bits recibidos con errores sobre el número total de bits transmitidos, el BLER mide la proporción de bloques con errores sobre el número total de bloques recibidos [21]. A medida que aumenta el BER, también aumenta la probabilidad de que un bloque contenga errores, lo que a su vez incrementa el BLER.

En este contexto, el BLER puede ser calculado de acuerdo a la siguiente expresión.

$$BLER = 1 - (1 - BER)^N \quad (2.3)$$

Donde:

- BLER: es la Tasa de Error de Bloque.
- BER: es la Tasa de Error de Bit.
- N: es el número de bits por bloque.

2.2. Inteligencia Artificial

La Inteligencia Artificial (IA) se refiere a la capacidad de una máquina para emplear algoritmos, aprender de los datos y tomar decisiones que se asemejan a las de un ser humano. Pero, a diferencia de los humanos, las máquinas incorporadas con IA pueden evaluar cantidades ingentes de datos a la vez. Además, la proporción de errores en las máquinas que realizan las mismas tareas que los humanos es significativamente menor. Dado que las capacidades de los ordenadores crecen exponencialmente con el tiempo, la idea de que son capaces de aprender y emitir juicios es algo que deberíamos considerar detenidamente. Gracias a estas capacidades, los sistemas de inteligencia artificial pueden realizar muchas actividades que antes sólo eran posibles para los humanos.[22]

2.2.1. Machine Learning

El Machine Learning (ML), conocido en español como aprendizaje de máquina o aprendizaje automático, es una disciplina de la informática y una rama de la inteligencia artificial (IA) que busca desarrollar sistemas capaces de aprender, reconocer patrones y predecir comportamientos a partir de conjuntos de datos [23].

Dentro de las herramientas más importantes de ML se encuentran las redes neuronales artificiales, conocidas en inglés como (Artificial Neural Networks (ANN)), y el aprendizaje profundo (Deep Learning (DL)).

2.2.2. Modalidades del Aprendizaje Automático

2.2.2.1. Aprendizaje Automático Supervisado

Es una de las técnicas más ampliamente estudiadas, debido a que se han desarrollado diversos métodos para su implementación. Este algoritmo, a partir de un conjunto de datos etiquetados de entrenamiento, determina una función que mapea a las diferentes permutaciones de las entradas una posible salida. Esto le permite generar resultados acertados incluso en escenarios no considerados durante el entrenamiento, es decir, en situaciones que no han sido incluidas en los datos etiquetados de entrenamiento [1].

2.2.2.2. Aprendizaje Automático no Supervisado

Al igual que en el Aprendizaje Automático Supervisado, el Aprendizaje Automático No Supervisado utiliza un conjunto de datos de entrenamiento, pero sin etiquetas que guíen al algoritmo durante su entrenamiento. Sin embargo, al procesar los datos, se pueden identificar similitudes o patrones que permiten representar el estado latente de los datos de entrada. Esto facilita la detección de anomalías, la predicción de estados futuros y la identificación de patrones de agrupamiento, entre otras aplicaciones[1].

2.2.2.3. Aprendizaje Automático por Refuerzo

El aprendizaje por refuerzo se sitúa entre el aprendizaje supervisado y el no supervisado, ya que utiliza señales de recompensa para guiar el aprendizaje, pero la información proporcionada es menos completa que en el aprendizaje supervisado. Esto es fundamental en problemas donde la solución óptima no es conocida y el agente debe aprenderla a través de su experiencia. [1].

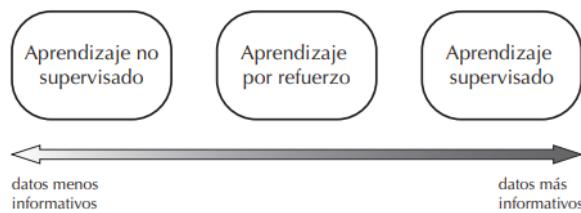


Figura 2.2: Distintos tipos de aprendizaje en función de los datos. Fuente: [1]

2.2.3. Deep Learning

El DL, conocido en español como aprendizaje profundo, es una rama del aprendizaje automático, se caracteriza por el uso de redes neuronales con múltiples capas y una gran cantidad de parámetros. Estas redes, también conocidas como redes neuronales profundas, son la base de la mayoría de los métodos en este campo [24]. Básicamente, el aprendizaje profundo intenta modelar datos a gran escala mediante múltiples capas de procesamiento con estructuras complejas [2]. Las capas iniciales tienden a aprender características simples, mientras que las capas posteriores se especializan en características más complejas, construidas a partir de las anteriores. Esta estructura jerárquica permite la construcción de representaciones de características potentes y detalladas [24]. Por tanto, a diferencia de las arquitecturas de aprendizaje

automático superficial, las arquitecturas de aprendizaje profundo se componen de múltiples transformaciones no lineales, como se muestra en la figura 2.3 [2].

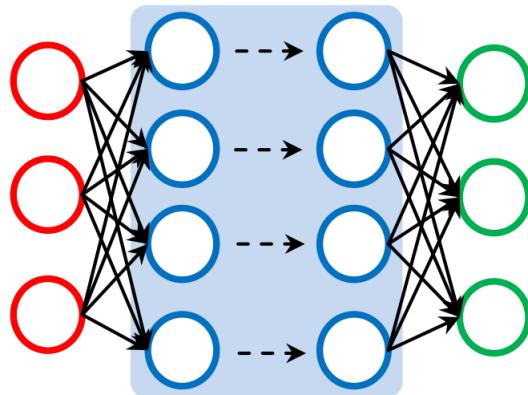


Figura 2.3: Red Neuronal Profunda con Estructura básica. Fuente: [2]

2.3. Redes de Neuronas Artificiales

Una red neuronal artificial es un modelo computacional que pertenece al campo del DL. Se inspiran en las funciones de las redes neuronales biológicas y se pueden definir como estructuras formadas por elementos adaptativos simples capaces de procesar datos en paralelo y representar conocimiento a partir de grandes cantidades de datos [23]. Las redes neuronales artificiales asignan entradas a salidas a través de relaciones no lineales, las cuales, mediante un proceso de entrenamiento, realizan tareas de clasificación y regresión [23]. Existen muchos tipos de arquitecturas de redes neuronales artificiales, pero todas comparten un elemento básico: las neuronas artificiales.

La arquitectura en la que las neuronas se conectan mantiene una relación directa con los algoritmos que pueden utilizarse para su entrenamiento.

2.3.1. Perceptrón

Un perceptrón es una neurona artificial que realiza cálculos para detectar características o tendencias en los datos de entrada. Su estructura consiste en múltiples entradas, una salida y una sola capa de pesos entre ellas, como se muestra en la figura 2.4 [3].

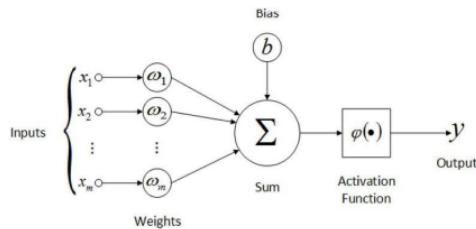


Figura 2.4: Estructura del Perceptrón. Fuente: [3]

El proceso de aprendizaje consiste en encontrar los valores óptimos para los pesos entre la capa de entrada y la de salida. Matemáticamente es útil pensar en las capas de entrada y de salida como vectores de valores (I y O respectivamente), y los pesos como una matriz W_{io} de dimensión $i \times o$, donde i es el número de neuronas de entrada, y o es el número de neuronas de salida. La salida de la red se calcula como se indica a continuación [3].

$$O = f \left(\sum_{i=1}^N I_i w_i + b \right) \quad (2.4)$$

Los datos en la capa de entrada son multiplicados por la matriz de pesos, el resultado de esta multiplicación es procesado por las neuronas de la capa de salida, utilizando una función de activación que determina si el nodo de salida se activa o no. La función de activación implementada en el modelo del perceptrón de 1940 se modulo de la siguiente manera [3]:

$$f(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{de otro modo} \end{cases} \quad (2.5)$$

El proceso de encontrar los valores correctos para los pesos se efectúa utilizando un paradigma de aprendizaje automático (entrenamiento), lo cual implica inicializar la matriz de pesos con un conjunto de números aleatorios entre -1 y +1. Posteriormente, a medida que el aprendizaje avanza, estos pesos se actualizan hasta que se decide que la red ha resuelto el problema. Durante la fase de entrenamiento, se evalúa la diferencia entre la suposición hecha por la red y el valor correcto para la salida, y los pesos se ajustan para minimizar el error. La técnica de minimización del error se basa en técnicas tradicionales de descenso de gradiente [3].

2.3.2. Redes Monocapa feedforward

En las redes neuronales más básicas y simples, los datos ingresan a través de una capa de entrada y se transmiten directamente a una capa de neuronas de salida sin retroalimentación, es decir, la capa es estrictamente del tipo hacia adelante (feedforward). Esta estructura se conoce como red neuronal monocapa, nombrando así principalmente la capa de salida, ya que la capa de entrada no realiza cálculos adicionales durante su operación [9].

2.3.3. Redes Multicapa feedforward

Este tipo de redes neuronales se caracteriza por tener una o más capas ocultas que realizan cálculos influenciando la salida final. El término *capas ocultas* se refiere a que no son directamente accesibles desde la capa de entrada o de salida. Estas capas permiten a la red neuronal abstraer patrones complejos presentes en los datos de entrada. En las redes neuronales multicapa, cada capa opera de manera secuencial: los datos de entrada se procesan en la primera capa oculta, luego se transmiten a la siguiente capa oculta y así sucesivamente, hasta llegar a la capa de salida.[9].

2.3.4. Redes Neuronales Recurrentes

Las redes neuronales recurrentes son diferentes de las redes feedforward porque al menos una de sus salidas se usa como entrada en la siguiente iteración. Además, no tienen una distinción clara entre capa de entrada y capa de salida como las redes feedforward. Estas redes se pueden clasificar en directas e indirectas dependiendo de cómo se estructuran sus conexiones recurrentes. [9]

2.3.5. Auto codificador

Los auto codificadores, Autoencoder (AE), son tipos de redes neuronales en las cuales la red se entrena para aprender a generar en la salida el mismo dato que se proporciona como entrada. Esta estructura consta de un codificador (encoder), encargado de transformar la entrada en una representación de menor dimensión, extrayendo las características esenciales que permiten su posterior reconstrucción en el decodificador (decoder) [15]. Por lo general, las capas ocultas tienen menos neuronas que las capas de entrada y salida, logrando así una representa-

ción de menor dimensión de los datos de entrada, es decir una representación comprimida [15].

La retropropagación (backpropagation) es un algoritmo que se utiliza para entrenar redes neuronales artificiales con el objetivo de reducir los errores durante el aprendizaje [15]. Los auto codificadores se entranan utilizando técnicas similares a las de una red neuronal tradicional, mediante el método de retropropagación [15]. En los modelos de aprendizaje automático, los parámetros son las variables que se estiman durante el proceso de entrenamiento con los conjuntos de datos. Los hiperparámetros de un modelo son los valores de las configuraciones utilizadas durante el proceso de entrenamiento. Por lo tanto, al diseñar un auto codificador, hay tres hiperparámetros fundamentales a considerar:

1. Número de capas: Depende fundamentalmente del número de capas ocultas, el cual puede variar de acuerdo a la complejidad requerida.
2. Número de neuronas por capa: Generalmente, el número de neuronas disminuye o se mantiene constante a medida que se avanza en las capas del codificador. Luego, se restablece en el orden inverso para el decodificador. El espacio latente, en un AE es la capa de cuello de botella donde se comprime y codifica la información de manera eficiente. En cuanto al tamaño de las neuronas en el espacio una menor dimensión implica una mayor compresión.
3. Función de pérdida: El error cuadrático medio es comúnmente utilizada, aunque depende de la arquitectura específica de cada auto codificador.

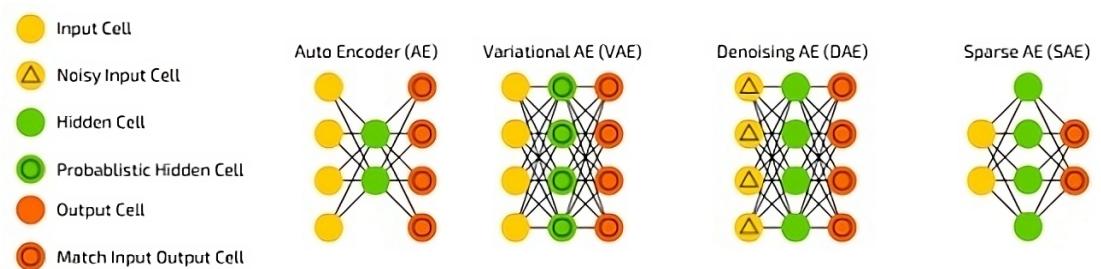


Figura 2.5: Tipos de auto codificadores. Fuente: [4]

2.3.5.1. Auto codificador variacional

Auto codificador Variacional, Variational Autoencoder (VAE) , por sus siglas en inglés, Variational autoencoder, permiten la codificación de los datos de entrada en un espacio latente, espacio de baja dimensión o espacio comprimido, formado por distribuciones de probabilidad

más sencillas, y la reconstrucción, a partir de dichas variables latentes, de los datos de origen [25]. A diferencia de las AE, comprimen las probabilidades en lugar de las características. Es posible que el VAE produzca nuevos datos que se parezcan a la distribución de origen. Esto se consigue codificando las características distintivas de la distribución de probabilidad representada por la muestra de población del conjunto de datos original como variables latentes [25]. Son capaces de producir nuevos ejemplos que siguen aproximadamente esa distribución utilizando el modelo aprendido.

2.3.5.2. Auto codificador de atenuación de ruido

Auto codificador de atenuación de ruido, Denoising Autoencoder (DAE), es generalmente clasificado como un tipo de red neuronal profunda. El DAE se entrena para usar una capa oculta para reconstruir un modelo limpio a partir de una versión dañada mediante técnicas de eliminación de ruido. Los auto codificadores de eliminación de ruido constituyen una técnica popular para eliminar el ruido no deseado de los conjuntos de datos, lo que permite realizar análisis precisos y fiables [25]. Son una herramienta muy eficaz en el campo de la programación informática, el procesamiento digital de señales y la ciberseguridad.

2.3.5.3. Auto codificador poco denso

Auto codificador poco denso, Sparse Autoencoder (SAE), es otro tipo de auto codificador que en algunos casos puede revelar algunos patrones de agrupación ocultos en los datos. La estructura es la misma que en el AE, pero el número de celdas ocultas es mayor que el número de celdas de la capa de entrada/salida. Un SAE busca que únicamente un pequeño número de neuronas en la capa oculta (la capa de representación) estén activas en cualquier momento dado [25]. Esto se logra mediante la introducción de un factor de penalización, en la función de pérdida del modelo, para limitar la activación no utilizada. Como resultado, el auto codificador disperso aprende a activar solamente un subconjunto limitado de unidades en la capa oculta, creando representaciones dispersas de los datos de entrada [25].

2.4. Arquitectura de las Redes de Neuronas Artificiales

2.4.1. Funciones de Activación

Las funciones de activación se usan para propagar la salida de las neuronas de una capa hacia la siguiente capa. Se tratan de funciones que producen la activación de la neurona, permitiendo además incorporar no linealidad al modelado de los datos de entrada de la red. La introducción de no linealidad permite a las redes neuronales aproximar funciones más complicadas y capturar patrones complejos en los datos de entrada. La función de activación se aplica luego de la operación lineal que se realiza en cada capa como se mostró en la ecuación 2.4.

2.4.1.1. Sigmoidal

Esta función de activación convierte cualquier valor de entrada en un número entre 0 y 1 en la salida. Principalmente se usa en la clasificación binaria debido a su limitación: tiende a saturarse, es decir, produce una salida cercana a 1 cuando la entrada es alta y cerca de 0 cuando es baja. Esto causa que durante el entrenamiento, los gradientes calculados sean muy pequeños, lo que dificulta que el algoritmo converja correctamente [26].

$$f(x) = \frac{1}{1 - e^{-x}} \quad (2.6)$$

2.4.1.2. Tangente hiperbólica

Esta función tiene un comportamiento parecido a la función sigmoide, pero con la diferencia de que los valores de salida están entre -1 y 1. Aunque también puede saturarse, una ventaja es que produce una salida simétrica, lo que hace más fácil entrenarla [26]. Sin embargo, el problema de la saturación limita claramente la aplicabilidad de esta función de activación en redes neuronales.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.7)$$

2.4.1.3. Rectified Linear Unit

La función de activación ReLU generará una salida de cero cuando la entrada sea negativa, y la salida será igual a la entrada cuando ésta sea positiva. En los últimos años, Rectified Linear Unit (ReLU) se ha convertido en la función más utilizada en modelos de Deep Learning,

principalmente debido a que no sufre de saturación como ocurre con las funciones sigmoide y tangente hiperbólica.[26]. En consecuencia, el algoritmo del gradiente descendente converge mucho más rápidamente, facilitando así el entrenamiento y es más fácil de implementar computacionalmente en comparación con las otras dos funciones.

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases} \quad (2.8)$$

2.4.1.4. SoftMax

La función Softmax transforma las salidas en una representación de probabilidades, asegurando que la suma de todas estas probabilidades sea igual a 1. La función SoftMax se utiliza en la clasificación multiclase debido a que asigna una probabilidad a cada clase. Esto significa que la función SoftMax es útil en escenarios donde se necesita saber la probabilidad de que una instancia pertenezca a cada clase; pero tiene limitaciones como la falta de adecuación para problemas de clasificación binaria [26].

$$f(x)_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \quad (2.9)$$

2.4.2. Algoritmos de optimización

Al trabajar con redes neuronales, el enfoque principal durante el entrenamiento es optimizar una métrica P que refleje el rendimiento de la red. La optimización de P se realiza de manera indirecta, a través de otra función llamada L . Esta función de pérdida L se diseña de tal manera que al optimizarla, también se mejore la métrica de rendimiento P [27].

El objetivo fundamental de un algoritmo de optimización radica en la reducción del error de generalización, una meta que podemos expresar como:

$$R(\theta) = E_p[L(f(x, \theta), y)] \quad (2.10)$$

Donde P representa la distribución de los pares (x, y) , L es la función que evalúa la pérdida para cada par, y $f(x, \theta)$ es la salida de la red neuronal. Esta cantidad simplemente corresponde a la pérdida asociada con un conjunto específico de parámetros θ , razón por la cual a menudo se le denomina riesgo". Por lo general, no tenemos conocimiento directo de P , pero

disponemos de un conjunto de entrenamiento. Por tanto, la estrategia consiste en minimizar el costo sobre este conjunto en lugar de hacerlo sobre la distribución completa. De esta manera, se busca minimizar el riesgo empírico. [27]

$$J(\theta) = E_{\hat{p}}[L(f(x, \theta), y)] = \frac{1}{n} \sum_{k=1}^n L(f(x_k, \theta), y_k) \quad (2.11)$$

2.4.2.1. Descenso de gradiente Estocástico

Los algoritmos de optimización que trabajan con la totalidad del conjunto de entrenamiento suelen ser clasificados como deterministas, en contraste, aquellos que operan con un subconjunto reducido de ejemplos (minilote) son conocidos como métodos estocásticos. Un algoritmo clásico de este tipo es el Stochastic gradient descent (SGD).[27]

El SGD se destaca como el algoritmo más eficaz para entrenar redes neuronales artificiales. En este contexto, los pesos del modelo representan los parámetros a ajustar, mientras que la función objetivo de pérdida mide el error de predicción promediado sobre un subconjunto (lote) de los datos de entrenamiento completos. La esencia del SGD radica en la noción de que el gradiente se puede aproximar mediante el valor esperado, y este último puede calcularse de manera eficiente utilizando un pequeño conjunto de muestras.[27]

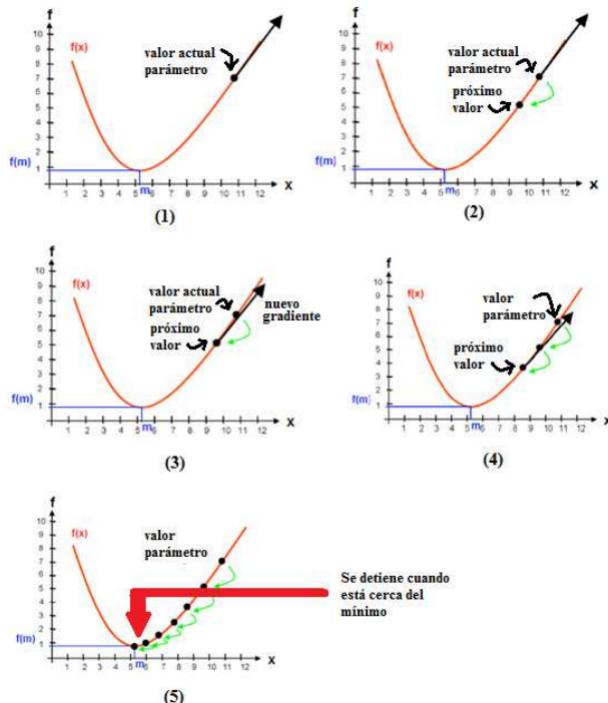


Figura 2.6: Funcionamiento del Descenso del Gradiente Estocástico. Fuente: [5]

El algoritmo fundamentalmente realiza lo siguiente, extrae una muestra de m elementos del conjunto de entrenamiento y calcula el estimado del gradiente de acuerdo a la siguiente expresión.

$$g = \frac{1}{m} \sum_k^m \nabla_{\theta} L(f(x_k, \theta), y_k) \quad (2.12)$$

La estimación del gradiente permite calcular el valor de θ , como $\theta = \theta - \eta g$, donde, θ representa el conjunto de pesos de la red neuronal que se actualizan durante el proceso de entrenamiento y η es la tasa de aprendizaje (*learning rate*) que controla el tamaño de los pasos que se dan en la dirección del gradiente negativo durante la actualización de los pesos. Este proceso se repite iterativamente hasta encontrar el valor de $\theta = \theta_0$. Es importante tener en consideración que la actualización de parámetros se realiza en cada capa, por lo que este proceso será iterativo y su complejidad depende del número de capas ocultas disponibles. [27]

Hasta este punto, hemos empleado una tasa de aprendizaje constante, pero debido a la variabilidad que se introduce al estimar el gradiente, con frecuencia es necesario reducirla de manera gradual. Así, adoptamos una tasa de aprendizaje η_k , donde k indica el número de iteración [27]. Las condiciones suficientes para asegurar la convergencia del SGD, conocidas como las condiciones de Robbins-Monro, se expresan como:

$$\sum_{k=1}^{\infty} \eta_k = \infty \quad (2.13)$$

$$\sum_{k=1}^{\infty} \eta_k^2 < \infty \quad (2.14)$$

Una característica relevante del algoritmo SGD y sus variantes es que el tiempo de cálculo por actualización de pesos no se incrementa con el número de conjuntos de entrenamiento. Aunque el número de actualizaciones necesario para alcanzar la convergencia tiende a aumentar con el tamaño del conjunto de entrenamiento, a medida que dicho tamaño se aproxima a infinito, el modelo eventualmente converge hacia su mínimo error óptimo sin necesidad de pasar por cada ejemplo del conjunto.

2.4.2.2. Descenso de gradiente Estocástico más momentum

El momentum o momento (γ) es un parámetro que permite acelerar el descenso de gradiente estocástico SGD.

$$a_{n+1} = \gamma a_n - \eta \nabla F(a_n) \quad (2.15)$$

Donde a_n representa el punto de la función y η el parámetro de aprendizaje. Cuando se utiliza el momento, esencialmente se está aumentando la velocidad de descenso hacia el mínimo, haciéndolo cada vez más rápido. Esto facilita una convergencia más rápida mientras reduce la oscilación durante el proceso de aprendizaje.

2.4.2.3. Gradiente acelerado Nesterov

Nesterov-accelerated Adaptive Moment Estimation (Nadam) es un parámetro que permite controlar el momento de una manera más inteligente. Este algoritmo ajusta la velocidad de descenso en función de la pendiente del gradiente. Por ejemplo, si se detecta un incremento en la magnitud del gradiente, Nadam ralentiza el momento para evitar pasos excesivos que podrían causar oscilaciones o divergencia [28].

De esta manera Nadam le da al momento esa capacidad de predicción. Si se adiciona esta mejora en la ecuación anterior se obtiene una aproximación para la siguiente posición, resultando en una convergencia más rápida y estable del modelo [28].

$$a_{n+1} = \gamma a_n - \eta \nabla F(a_n - \gamma a_n) \quad (2.16)$$

Esta capacidad de anticipación evita que el descenso de gradiente se realice demasiado rápido, proporcionando una mayor estabilidad y mejorando el rendimiento de las redes neuronales.

2.4.2.4. Adagrad

Adaptive Gradient Algorithm (Adagrad) ajusta la velocidad de aprendizaje mientras se entrena, usando la suma de gradientes previos. Esto significa que actualiza más intensamente para datos menos comunes y con menor intensidad para datos más frecuentes. La velocidad de aprendizaje se calcula basándose en la raíz cuadrada media (RMS) de los gradientes previos. [28].

2.4.2.5. Adadelta

Adaptive Delta Gradient (Adadelta) es una versión avanzada de Adagrad que ajusta su tasa de aprendizaje de manera más suave. En lugar de almacenar todos los gradientes anteriores, utiliza una ventana que acumula solo los gradientes más recientes. [28].

2.4.2.6. RMSProp

Root Mean Square Propagation (RMSprop) es un método de optimización diseñado para abordar los problemas de la disminución rápida en Adagrad. Reconoce que simplemente acumular gradientes no es suficiente. RMSprop introduce la idea de amortiguar las fluctuaciones en los gradientes. En lugar de acumular todos los gradientes, utiliza una media móvil exponencial (EWMA) que automáticamente descarta valores muy antiguos.[28].

2.4.2.7. Adam

Adaptive Moment Estimation (Adam) es un algoritmo de optimización que combina los conceptos del RMSprop y el momentum para mejorar el proceso de aprendizaje de un modelo. Adam usa una estimación del momento y magnitud de los gradientes previos para actualizar los parámetros del modelo en cada iteración; sin embargo, la tasa de aprendizaje de cada parámetro individualmente es adaptativa. Esto permite que el modelo se ajuste de manera más eficiente a los datos de entrenamiento, lo que puede resultar en predicciones más precisas en comparación con otros métodos de optimización.[29].

2.4.3. Funciones de pérdida

Las funciones de pérdida, también conocidas como funciones objetivo o criterios de error, son utilizadas para cuantificar qué tan bien un modelo de aprendizaje automático está realizando una tarea específica. Estas funciones toman las predicciones del modelo y las comparan con los valores reales para calcular un valor que representa la pérdida o error. Son herramientas matemáticas que ayudan al algoritmo de optimización a ajustar los parámetros del modelo durante el proceso de entrenamiento para minimizar el error [30].

La elección de la función de pérdida depende del tipo de problema de aprendizaje automático que se esté abordando. A continuación se presenta algunas de las funciones de pérdida más

empleadas, dependiendo de si se trata de problemas de regresión o de clasificación.

2.4.3.1. Error Cuadrático Medio (MSE)

Los problemas de regresión suelen implicar el error cuadrático medio, a veces conocido como MSE (Mean Squared Error). Para calcularlo se utiliza la media de las diferencias al cuadrado entre los valores esperados y los reales; es decir:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 \quad (2.17)$$

Donde hay n muestras, $y^{(i)}$ son los valores respuesta teóricos e $\hat{y}^{(i)}$ son los valores respuesta obtenidos. El problema de determinar si el valor teórico es inferior al valor obtenido se elimina calculando de esta manera. Sin embargo, los errores importantes están sujetos a sanciones importantes. La convexidad del error cuadrático medio se debe a su mínimo global bien definido. El principal inconveniente es su sensibilidad a los valores atípicos, lo que puede aumentar considerablemente el error si el valor predicho es significativamente menor que el valor teórico.[30].

2.4.3.2. Error Absoluto Medio (MAE)

El error absoluto medio o MAE (Mean Absolute Error) es una función de pérdida alternativa al error cuadrático medio, también empleada en problemas de regresión. Calcula la diferencia en valor absoluto entre los valores respuesta teóricos y los obtenidos, es decir:

$$MSE = \frac{1}{n} \sum_{i=1}^n |y^{(i)} - \hat{y}^{(i)}| \quad (2.18)$$

El error absoluto medio es menos susceptible a los valores atípicos, dado que, a diferencia de el error cuadrático medio, el factor es lineal, no cuadrático [30].

2.4.3.3. Entropía Cruzada Binaria

La entropía cruzada binaria es un aspecto importante en los problemas de clasificación. la expresión se define asumiendo que sólo hay dos resultados posibles en la clasificación y se utiliza:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n (y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})) \quad (2.19)$$

Es habitual emplearla en combinación con la función de activación ‘Sigmoide’ en la capa de salida de la red neuronal [30].

2.4.3.4. Entropía Cruzada Categórica

Para modelos en los que hay más de dos posibles resultados en la clasificación, se utiliza la entropía cruzada categórica (Categorical Cross Entropy). Su fórmula está definida por:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m y_{ij} \log \hat{y}^{(ij)} \quad (2.20)$$

Es habitual emplearla en combinación con la función de activación ‘SoftMax’ en la capa de salida de la red neuronal [30].

2.4.4. Técnicas de Regularización

El sobreajuste ocurre cuando un modelo hace predicciones exactas con los datos de entrenamiento pero no con nuevos datos. En Deep Learning, se utilizan métodos de regularización para mejorar la capacidad de generalización del modelo y evitar el sobreajuste. Aquí se mencionan algunos de los métodos más comunes:

2.4.4.1. Regularización de peso L1

Se trata de una forma de regularización de decaimiento de pesos, también conocida como regresión Lasso. Este método utiliza el valor absoluto de los términos en la función de penalización en lugar de sus cuadrados. Como resultado, algunas variables de entrada con características menos relevantes tienen una influencia mínima o nula en la salida, lo que conduce a una selección automática de variables. Esto ayuda al modelo a identificar y utilizar únicamente las características más importantes [31].

2.4.4.2. Regularización de peso L2

Es una forma diferente de regularización que también se conoce como regresión de crestas. Esta técnica es parecida a la regularización L1, pero en lugar de sumar los valores absolutos de los coeficientes, suma los cuadrados de estos coeficientes. Esto ayuda a obtener modelos con coeficientes más pequeños y reduce el riesgo de sobreajuste [31].

2.4.4.3. Dropout

En cada capa oculta de este método, se desactiva un porcentaje de neuronas durante el entrenamiento del modelo. El propósito es evitar que el modelo memorice los datos de entrada, lo cual podría llevar a sobreajustarlos. En cambio, se fuerza al modelo a establecer los pesos de las neuronas conforme aprende del contexto de los datos. Cada neurona se especializa en detectar ciertas características, y al desactivar algunas, las neuronas activas ajustan sus pesos para cubrir las características que las desactivadas no están considerando. Esto resulta en una mejor capacidad del modelo para generalizar cuando todas las neuronas están activas [31].

2.4.4.4. Aumento de Datos (Data Augmentation)

Esta técnica implica aplicar diversas transformaciones a las entradas para obtener datos ligeramente diferentes, que pueden considerarse como nuevas muestras del mismo tipo. Es ampliamente utilizada en visión artificial, donde transformaciones como voltear, recortar, deformar la perspectiva, ajustar el brillo, contraste y saturación, entre otras, generan versiones variadas de las imágenes originales. Esto enriquece el conjunto de entrenamiento al aumentar la diversidad de los datos, lo cual mejora la capacidad del modelo para generalizar y reduce el riesgo de sobreajuste. [31].

2.4.4.5. Parada Temprana (Early Stopping)

El early stopping, o parada temprana, es una estrategia que consiste en vigilar de manera continua el proceso de entrenamiento para determinar cuándo detenerlo antes de que ocurra sobreajuste o subajuste. Inicialmente, tanto el conjunto de datos de entrenamiento como el de validación suelen mejorar en cada época de entrenamiento. Sin embargo, llega un punto donde el conjunto de entrenamiento sigue mejorando mientras que el conjunto de validación comienza

a empeorar (sobreajuste). En ese momento, esta técnica guarda los resultados obtenidos hasta la fecha y detiene el entrenamiento para prevenir un aumento en el error [31].

2.4.4.6. Normalización por lotes (Batch Normalization)

En esta técnica, la función principal es eliminar la inestabilidad, que se puede introducir, durante el entrenamiento profundo de la red neuronal normalizando las activaciones de la salida de la capa [31]. Para esto se utiliza la media y la varianza obtenidos de todo el conjunto de datos de entrenamiento, o en caso de usar gradiente estocástico, la media y varianza de cada conjunto de entrada [31].

Es particularmente útil para entrenar redes muy profundas, ya que puede ayudar a reducir el cambio de covariable interno que puede ocurrir durante el entrenamiento. El término “cambio de covariable interno” se utiliza para describir el efecto que tiene la actualización de los parámetros de las capas superiores en la distribución de las entradas a la capa actual durante el entrenamiento de aprendizaje profundo. Esto puede dificultar el proceso de optimización y ralentizar la convergencia del modelo [32].

En este caso, la normalización se refiere al ajuste de la distribución de los datos para que tengan una media cercana a cero y una desviación estándar cercana a uno. La normalización por lotes es una técnica utilizada para mejorar el rendimiento de una red de aprendizaje profundo eliminando primero la media del lote y luego dividiéndola para la desviación estándar del lote [32].

$$x = \frac{x - \mu}{\sigma} \quad (2.21)$$

De esa manera, la salida de cada neurona sigue una distribución normal estándar en todo el lote. Sin embargo, forzar que todas las preactivaciones sean cero con una desviación estándar unitaria en todos los lotes puede resultar demasiado restrictivo, en el caso de que las distribuciones fluctuantes sean necesarias para que la red aprenda mejor ciertas clases [32].

Para solventar esta situación, la normalización por lotes introduce dos parámetros entrenables: γ y β , y permite al modelo elegir la distribución más adecuada para cada capa oculta, ajustando los valores óptimos de estos dos parámetros [32]:

- γ es un factor de escala que permite ajustar la desviación estándar;

- β es un factor de desplazamiento que permite ajustar el sesgo, desplazando la curva hacia la derecha o hacia la izquierda [32].

$$y_i = BN(x_i) = \gamma \hat{x}_i + \beta \quad (2.22)$$

A continuación, se muestra el proceso de cálculo que realiza el proceso de normalización por lotes. Siendo x la preactivación correspondiente a la k -ésima neurona en una capa y μ_B y σ_B^2 la media y varianza del mini-batch. El mini-batch es un conjunto de muestras fijo de entrenamiento y que es menor que el conjunto de datos real [32]:

$$\mu_B = \frac{1}{B} \sum_{i=1}^B Bx_i \quad (2.23)$$

$$\sigma_B^2 = \frac{1}{B} \sum_{i=1}^B (x_i - \mu_b)^2 \quad (2.24)$$

$$\hat{x}_i = \frac{x_i - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} \quad (2.25)$$

Se adiciona ϵ como ayuda para evitar una posible división para cero 0, cuando σ_B^2 es muy pequeña.

La normalización por lotes se puede usar para regularizar debido al uso del *momentum*, el cual consiste en tener en cuenta un registro de uso de la media y la desviación de cada mini-batch de entrada, y obtener una ponderación entre los valores actuales y los anteriores [31]. Este proceso se realiza debido a que durante la prueba o inferencia es posible que no se cuente con un lote para realizar estos cálculos.[32]

2.5. Reducción de Dimensionalidad

La reducción de la dimensionalidad consiste en transformar un conjunto de datos de alta dimensionalidad en un espacio de menor dimensionalidad, es decir, es el proceso mediante el cual se reduce la cantidad de variables o características en un conjunto de datos, manteniendo la información relevante y significativa. Esto se emplea principalmente para la compresión de los datos, la reducción del ruido o como etapa previa a la clasificación. Aunque también se emplea para la visualización de conjuntos de datos de alta dimensionalidad, los cuales serían

imposibles de representar gráficamente sin el uso de esta técnica, debido a que contienen un elevado número de atributos [33].

2.5.1. Análisis de Componentes Principales (PCA)

El método PCA, consiste en una técnica de reducción de la dimensionalidad lineal y no supervisada, se centra en encontrar las principales direcciones de variabilidad de los datos y de representarlos en un espacio de menor dimensionalidad consiguiendo resumir casi toda la información en unos pocos componentes. El procedimiento de cálculo es el que se describe a continuación [33]:

- Se requiere estandarizar los datos para que no presenten escalas diferentes, para así evitar que variables con escalas dominantes afecten demasiado la variabilidad en la dirección de los componentes principales.
- Luego, se calcula la matriz de covarianza; la cual captura las relaciones lineales entre las variables y proporciona información sobre la variabilidad y las dependencias entre ellas.
- A continuación, se calculan los autovectores y autovalores de la matriz de covarianza. Los autovectores representan las direcciones en las que los datos tienen la mayor varianza, y los autovalores indican la cantidad de varianza explicada por esa dirección. Los autovectores están ordenados por sus autovalores en orden descendente, lo que significa que los primeros autovectores capturan la mayor parte de la varianza en los datos.
- Para reducir la dimensionalidad, se seleccionan los primeros k autovectores correspondientes a los k autovalores más grandes. Estos autovectores se denominan componentes principales y forman una base ortogonal en el nuevo espacio de menor dimensionalidad.
- Finalmente, los datos originales se proyectan en el nuevo espacio de componentes principales, esto implica calcular el producto escalar entre los datos estandarizados y los autovectores seleccionados. El resultado de esta proyección es una representación de los datos en un espacio de menor dimensionalidad, donde cada componente principal es una combinación lineal de las variables originales.

2.5.2. t-Distributed Stochastic Neighbor Embedding (t-SNE)

La técnica de incrustación de vecinos estocásticos distribuidos en t (t-Distributed Stochastic Neighbor Embedding, t-SNE) es una herramienta de reducción de dimensionalidad no super-

visada y no lineal. Su principal objetivo es visualizar datos de alta dimensión en un espacio de baja dimensión, típicamente en dos o tres dimensiones, lo que facilita la interpretación y comprensión de los datos [34].

t-SNE funciona manteniendo cerca en el espacio de baja dimensión a aquellos puntos que están próximos en el espacio de alta dimensión. Para lograr esto, se calcula una matriz de similitudes entre los puntos en el espacio original. A partir de la matriz de similitudes, se construyen distribuciones de probabilidad que miden la similitud relativa entre los puntos. Estas similitudes entre puntos vienen determinadas por una distribución Gaussiana; suponiendo que x_i y x_j son puntos distintos en el espacio de alta dimensión, $p_{i|j}$ mide la cercanía entre x_j y x_i , considerando una distribución Gaussiana alrededor de x_i con una varianza σ_i^2 [33].

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)} \quad (2.26)$$

Posteriormente, se realiza el cálculo de similitudes, pero esta vez en el espacio de baja dimensionalidad, utilizando la distribución t de Student. Esta distribución cae más lentamente que la Gaussiana, permitiendo más margen y mitigando el problema del hacinamiento. El objetivo es encontrar una distribución de probabilidad que sea similar a la construida en el espacio original, es así que, $q_{i|j}$ define una matriz de similitud en el espacio de menor dimensión, donde y_i e y_j son puntos en el espacio de menor dimensión [33].

$$q_{i|j} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}} \quad (2.27)$$

Y finalmente, busca encontrar las proyecciones en el espacio de baja dimensionalidad que minimicen la divergencia entre las distribuciones de probabilidad del espacio original y del espacio proyectado [33]. El objetivo principal de la reducción de dimensionalidad es preservar la mayor cantidad posible de la estructura significativa de los datos originales en el nuevo espacio de baja dimensión [34].

2.5.3. Uniform Manifold Approximation and Projection (UMAP)

UMAP es otro algoritmo de reducción de la dimensionalidad no supervisado y no lineal. Su objetivo principal es preservar la estructura global y las relaciones locales entre los puntos en un espacio de menor dimensionalidad. Funciona de manera similar a t-SNE pero difiere en la forma en la que calcula las distancias entre puntos [33].

Primero, es importante definir el concepto matemático de variedad: una variedad d-dimensional es una parte de un espacio n-dimensional (donde $d < n$) que localmente se asemeja a un hipoplano de dimensión d. En otras palabras, las variedades son espacios topológicos que tienen similitudes locales con el espacio Euclídeo. El algoritmo UMAP parte de la idea de que los datos, originalmente distribuidos en un espacio de muchas dimensiones, en realidad residen en una variedad de dimensiones mucho más reducidas. A partir de esta premisa, UMAP calcula distancias entre puntos en esta variedad y busca una manera de representar esos datos en un espacio de dimensiones más bajas, manteniendo una estructura topológica similar. En otras palabras, UMAP conserva tanto la forma local como la estructura general de los datos durante el proceso de reducción de dimensiones. [35].

3. Trabajos Relacionados

Los avances recientes en el aprendizaje profundo han llevado a la aparición de sistemas de comunicación basados en auto codificadores, Autoencoder (AE). Las técnicas de aprendizaje profundo se pueden emplear para diseñar un sistema de comunicación de extremo a extremo utilizando un codificador para reemplazar las tareas del transmisor, como la modulación y la codificación, y un decodificador para las tareas del receptor, como la demodulación y la decodificación [11].

Este capítulo explora trabajos relacionados para la implementación de auto codificadores para sistemas de comunicación de extremo a extremo. En la Sección 3.1 se abordan trabajos relaciones con el diseño de la arquitectura y parámetros de diseño de auto codificadores. Mientras que en la Sección 3.2 se presentan algunas limitaciones y soluciones encontradas sobre las funciones de activación en las capas de entrada, salida y ocultas; así como en el entrenamiento del auto codificador.

3.1. Implementación de auto codificadores extremo a extremo

En [6], los autores introdujeron por primera vez el auto codificador como un nuevo diseño de sistema de comunicación. El sistema se presentó como un esfuerzo para que el transmisor y el receptor se optimizan en un solo proceso.

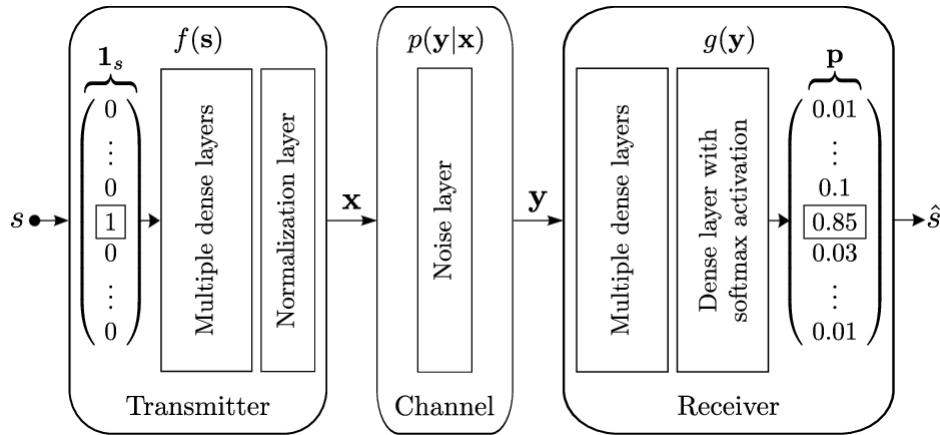


Figura 3.1: Un sistema de comunicaciones representado como un AE. Fuente: [6].

En la Figura 3.1 se puede ver un esquema general de la implementación realizada. El transmisor consta de una Feedforward Neural Networks (FNN) con múltiples capas densas seguidas de una capa de normalización que garantiza que se cumplan las restricciones físicas en x . La

entrada s al transmisor está codificada como un vector one-hot $1_s \in \mathbb{R}^M$, es decir, un vector M -dimensional, cuyo enésimo elemento es igual a uno y cero en caso contrario. El canal está representado por una capa de ruido aditivo con una varianza fija $\beta = (2RE_b/N_0)^{-1}$, donde E_b/N_0 denota la relación entre energía por bit (E_b) y densidad espectral de potencia de ruido (N_0). El receptor también se implementa con FNN, la última capa utiliza una activación softmax cuya salida es un vector de probabilidad, $p \in (0, 1)^M$, sobre todos los mensajes posibles [6].

El codificador automático se puede entrenar de un extremo a otro usando SGD en el conjunto de todos los mensajes posibles $s \in M$ usando la función de pérdida de entropía cruzada categórica muy adecuada entre 1_s y p . Finalmente, el mensaje decodificado \hat{s} corresponde al índice del elemento de p con mayor probabilidad [6].

Otro auto codificador se analiza en [14] donde el transmisor, canal y receptor emplean los mismos criterios descritos anteriormente; sin embargo, para el entrenamiento utilizaron diferentes optimizadores para actualizar los pesos de la FNN y comparar su comportamiento. En el proceso de entrenamiento se emplearon los optimizadores Stochastic gradient descent (SGD), Root Mean Square Propagation (RMSprop), Adaptive Gradient Algorithm (Adagrad), RMSprop, Adaptive Delta Gradient (Adadelta), Infinite norm based on Adam (Adamax), Adaptive Moment Estimation (Adam) y Nesterov-accelerated Adaptive Moment Estimation (Nadam). Los resultados demostraron que el codificador automático aprende el esquema de codificación y modulación optimizando conjuntamente la función de costo para todo el modelo de un extremo a otro. Sin embargo, el entrenamiento con entropía cruzada categórica y optimizado con Adadelta ofrece el mejor rendimiento en términos de Signal to noise ratio (SNR) frente al Block Error Rate (BLER).

En [36] utilizan secuencias de bits aleatorias para entrenar un auto codificador que minimiza la pérdida de error cuadrático medio entre los símbolos originales y los reconstruidos. Los datos de entrenamiento incluyen 10.000 símbolos y se utilizan 200 símbolos para la validación; se utilizó el optimizador Adam para el entrenamiento con un tamaño de mini-batch de 400 muestras y un máximo de 10 épocas. Cada 5 épocas, la tasa de aprendizaje inicial decrece un valor de 0,1.

El sistema fue verificado mediante el envío de texto e imágenes empleando una modulación de hasta 4 símbolos. Para la transmisión de texto, cada carácter de los datos de texto está representado por un código ASCII de 7 bits, que después de una única codificación se obtienen 128 vectores binarios únicos. De esta manera, una capa completamente conectada con 128 nodos de entrada y M nodos de salida actúa como un codificador, mientras que el decodificador

se representa utilizando M nodos de entrada y 128 nodos de salida. Para la transmisión de imágenes, primero son convertidas a imágenes en escala de grises y los valores de píxeles se normalizan para tener valores entre 0 y 255. El codificador y el decodificador utilizan una lógica similar. Los resultados apuntan al modelo de auto codificador como una opción más viable para la comunicación inalámbrica de extremo a extremo que utiliza técnicas de aprendizaje profundo, ya que muestra mejor capacidad de adaptarse a diversas entradas y mantener una comunicación confiable debido a sus bajos valores de Bit Error Rate (BER).

3.2. Limitaciones en la implementación de auto codificadores

3.2.1. Tendencia a cero o infinito del gradiente

Las redes recurrentes se enfrentan el problema de que los pesos se incrementen hasta el infinito en el aprendizaje. En las redes alimentadas hacia adelante el problema es que los pesos tienden a cero, especialmente al usar funciones de activación que saturan en valores pequeños [7]. En redes neuronales profundas entrenadas utilizando algoritmos basados en retropropagación de gradiente, el aumento de peso de la primera capa generalmente tiende a cero[7]. Esto se muestra en la siguiente ecuación cuando los pesos, w_{ki} , son valores muy pequeños:

$$\delta_{ip} = F'(net_{ip}) \sum_k w_{ki} \delta_{kp} \quad (3.1)$$

La suma ponderada de los gradientes de la capa siguiente y la derivada de la función de activación de la neurona actual conforman esta función. El problema se agrava en el caso de las funciones sigmoides (\tanh y logística), ya que su derivada tiene valor cero para valores ínfimos. Este problema se resuelve con la función ReLU, cuya derivada es siempre 1 para las entradas positivas y 0 para las negativas. Los pesos se hacen cada vez más pequeños a medida que aumenta el número de capas ocultas porque la segunda componente hace que el valor de los pesos cercano a cero se propague hacia atrás. [7], [26].

3.2.2. Sobreajuste

Para nuevos valores, diferentes al conjunto de entrenamiento, la red debe ser capaz de predecir con precisión los valores de salida, esto se conoce como generalización. El sesgo (bias), también conocido como error de entrenamiento, cuantifica la diferencia entre las predicciones del modelo de entrenamiento y el resultado previsto. Los resultados imprecisos del conjunto

de entrenamiento están implícitos en un sesgo grande. De forma similar, la varianza calcula la diferencia entre los errores del conjunto de prueba y del conjunto de entrenamiento. Una variación grande indica resultados inexactos para vectores de datos nuevos, o nunca antes evaluados [26].

El cálculo de la varianza puede realizarse durante las fases del entrenamiento utilizando el error de la red para el conjunto de pruebas, y hacer un cálculo final utilizando el error del conjunto de validación en su lugar. Debido al compromiso varianza/sesgo, es difícil determinar la fuente del error y solucionarlo, pero una estrategia habitual para medir el sobreajuste en tiempo real durante el entrenamiento consiste en dividir el conjunto de vectores de entrenamiento en tres conjuntos de datos: conjunto de entrenamiento, conjunto de pruebas y conjunto de validación [26].

El error de red del conjunto de prueba puede utilizarse para calcular la varianza durante las fases de entrenamiento, y el error del conjunto de validación puede utilizarse para el cálculo final. Aunque es difícil identificar la causa del error y abordarlo debido al equilibrio entre varianza y sesgo, dividir el conjunto de vectores de entrenamiento en tres conjuntos de datos: el conjunto de entrenamiento, el conjunto de prueba y el conjunto de validación es un método para cuantificar el sobreajuste durante el entrenamiento.

Una técnica común de aprendizaje profundo es el entrenamiento por lotes (batch); esta técnica genera mejores resultados en tiempo y reduce el problema del sobreajuste. El algoritmo calcula los incrementos de peso y los acumula sin cambiar los pesos de conexión analizando un lote de n vectores de datos de entrenamiento. Estos se actualizan utilizando el total de los incrementos de cada peso solo cuando se completa la evaluación de todo el conjunto.

3.2.3. Conclusiones

Actualmente los algoritmos de entrenamiento utilizados están basados en el de retropropagación de gradiente. Sin embargo, el entrenamiento de redes neuronales conlleva un elevado coste computacional, un elevado número de operaciones para obtener los pesos óptimos de la red. Una posible solución es la implementación de una red neuronal eficiente, donde el coste de almacenamiento y entrenamiento no sea excesivo.

Se han desarrollado numerosos métodos orientados a reducir los problemas de tendencia a cero o infinito del gradiente, como la función ReLU, cuya derivada siempre es 0 para valores negativos y 1 para positivos, eliminando el problema que se producía al utilizar derivadas

de funciones que se saturan. Las funciones de inicialización, así como diferentes métodos de pre-entrenamiento buscan encontrar buenos puntos de partida dentro del espacio de búsqueda de pesos. Los métodos de inicialización de He y Xavier obtienen buenos resultados y, aunque aumentan el tiempo de inicialización, este es despreciable en comparación con el de entrenamiento, puesto que se realiza una única vez al comienzo del algoritmo.

Se han ideado muchas técnicas para atenuar los problemas de un gradiente con tendencia a cero o al infinito. Una de ellas es la función Rectified Linear Unit (ReLU), cuya derivada es siempre 0 para los valores negativos y 1 para los positivos, con lo que se elimina el problema que surge al emplear derivadas de funciones que se saturan. Además, varias técnicas de pre-entrenamiento y funciones de inicialización buscan puntos de inicio adecuados en el espacio de búsqueda de pesos. El aumento del tiempo de inicialización de He y Xavier es mínimo en comparación con el tiempo de entrenamiento, porque sólo hay que hacerlo una vez al principio del algoritmo, pero aun así producen buenos resultados.

4. Diseño e Implementación el auto codificador

Este capítulo proporciona detalles acerca de la arquitectura del auto codificador, incluyendo información sobre los elementos, parámetros y consideraciones necesarias en el diseño e implementación. Este capítulo se estructura de la siguiente manera. En la sección 4.1 se presentan las consideraciones que justifican la arquitectura seleccionada. La Sección 4.2 introduce detalles del diseño e implementación del lado del codificador, describiendo los datos de entrada, capa de entrada, capas ocultas o densas y la capa de normalización. A continuación, la Sección 4.3 presenta el modelo de canal empleado para el auto codificador. La Sección 4.4 describe el diseño e implementación del lado del decodificador enfocándose en las capas densas y los datos de salida. Finalmente, la sección 4.5 muestra como generar y configurar el modelo. También se describen las consideraciones a tener en cuenta en el entrenamiento para evitar las deficiencias descritas en el Capítulo 3.

4.1. Análisis previo para la arquitectura

El problema de clasificación es un problema estadístico o de aprendizaje automático que tiene como objetivo asignar una categoría o etiqueta a un conjunto de datos de entrada en función de sus propiedades o características. El objetivo básico es aprender un mapeo entre los datos de entrada y categorías o clases predefinidas, y luego usar este mapeo para predecir nuevas etiquetas de clase para los nuevos datos.

En el contexto de un auto codificador (Autoencoder (AE)), este enfoque de aprendizaje se puede adaptar a la tarea específica de reconstruir los puntos de datos de entrada en lugar de predecir directamente la clase a la que corresponden dichos puntos. Este proceso implica comprimir los datos de entrada en una representación de baja dimensión (codificación) y luego reconstruir los datos originales a partir de esa representación (decodificación). Si el AE se entrena con una alta precisión, la calidad de la reconstrucción es un indicador de la capacidad del modelo para comprender y generalizar los patrones encontrados en los datos de entrada. Este concepto aplicado a un sistema de comunicación, es valioso para reducir el efecto del ruido White Gaussian Noise (WGN) y mejorar la probabilidad de error del sistema.

En este contexto, la Figura 4.1 proporciona la arquitectura general de un auto codificador aplicado a un sistema de comunicaciones, el cual depende de la cantidad de símbolos del esque-

ma de modulación a utilizar. Por lo tanto, la capa de entrada y la de salida tienen las mismas dimensiones con el objetivo de que el auto codificador aprenda a reconstruir con precisión los datos de entrada proporcionados. Las capas densas del auto codificador suelen tener un tamaño más pequeño que las capas de entrada y salida, lo que les permite comprimir los datos.

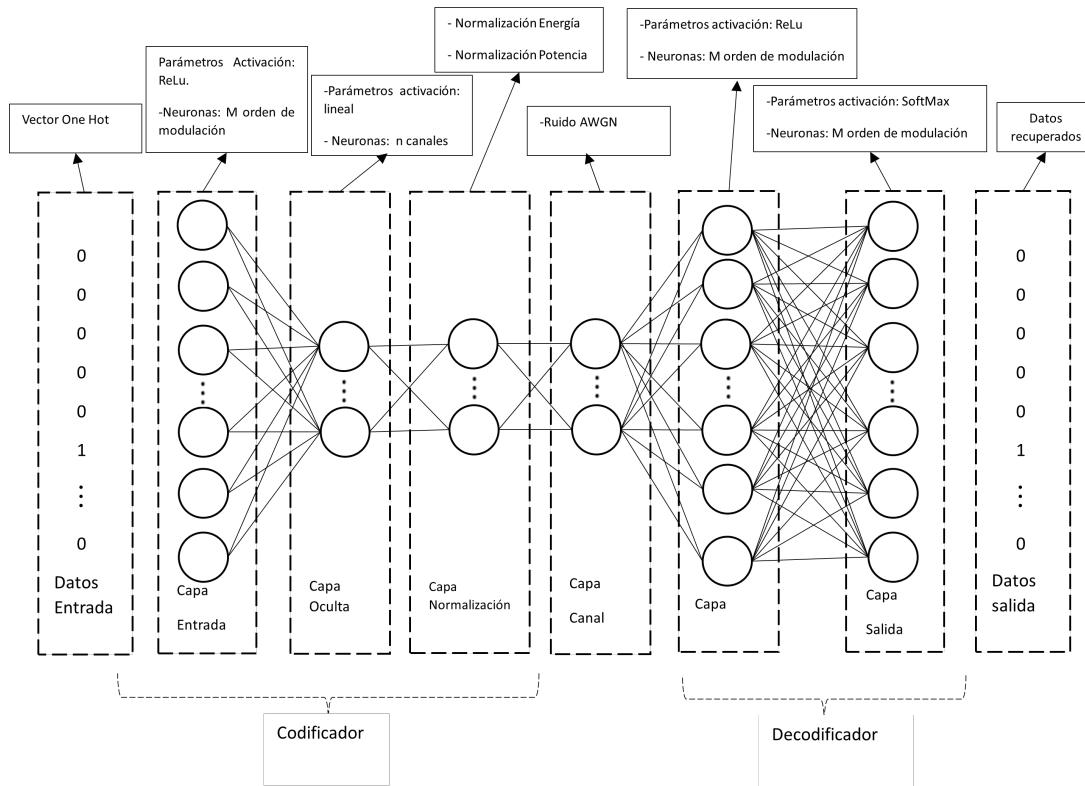


Figura 4.1: Propuesta de Esquema de auto codificador. Fuente: Autores.

4.2. Arquitectura para el Codificador

Se emplearon dos diferentes tipos de ordenadores para el diseño, codificación y pruebas de la red neuronal desarrollada en lenguaje Python utilizando las librerías de Tensorflow, Keras, OpenCV, Scipy y Numpy. Los archivos que contienen el código fuente implementado se puede encontrar en [37].

4.2.1. Datos de Entrada

La codificación conocida como *One Hot Encoding* se usa comúnmente para preprocesar características categóricas en modelos de aprendizaje automático. Este proceso es crucial para transformar variables categóricas en un formato comprensible por los algoritmos de apren-

dizaje automático, con el objetivo de mejorar la predicción del modelo y la precisión en la clasificación.

La Tabla 4.1 muestra un ejemplo de codificación *One Hot* para la modulación Quadrature Phase Shift Keying (QPSK). Esta técnica consiste en crear una variable binaria para cada categoría de la variable categórica. Una categoría está presente cuando su valor es 1, y está ausente cuando su valor es 0. Utilizar la codificación *One Hot* es útil cuando no hay un orden particular entre las categorías y todas son igualmente importantes.

Tabla 4.1: Ejemplo de codificación *One Hot* para QPSK

ID	00	01	10	11
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Para una modulación de mayor orden, se realiza un procedimiento similar que implica agregar columnas en función del número de variables categóricas (es decir, el número de símbolos). En estas columnas, se coloca el valor de 1 en la nueva categoría, mientras que el resto de las columnas corresponden al valor de 0. En la Tabla A.1 del Anexo A.1 se muestra un ejemplo de esta codificación aplicada a 16-Quadrature Amplitude Modulation (16-QAM). Por ultimo, una forma de generalizar este tipo codificación se observa en la Tabla 4.2.

Tabla 4.2: Ejemplo de codificación generalizada *One Hot*

ID	símbolo 1	símbolo 2	símbolo 3	...	símbolo N
1	1	0	0	...	0
2	0	1	0	...	0
3	0	0	1	...	0
:	:	:	:	..	:
N	0	0	0	...	1

4.2.2. Capa de entrada

El modelo secuencial de Keras es simple, pero limitado en su topología; mientras que la API funcional es útil para crear modelos complejos de múltiples entradas y múltiples salidas. Esto lo hace ideal para casos con arquitecturas de modelos más complejas, como auto codificadores que requieren estructuras especiales de codificación y decodificación.

En un auto codificador, la capa de entrada es la parte del codificador que actúa como la primera capa del modelo que recibe los datos de entrada y los codifica en una representación interna. En esta capa, cada neurona representa una función o dimensión de los datos de entrada. Por lo tanto, el número total de neuronas en la capa de entrada es igual al número, M , de símbolos a transmitir. Es decir, cada neurona en la capa de entrada corresponde a una entrada del vector binario que representa una variable categórica de la codificación *One Hot*. En la API funcional de Keras, esta capa de entrada se define mediante la función *Input*, que especifica la forma de los datos de entrada del modelo.

Extracto de código 4.1: Implementación de capa de entrada

```
input_signal = Input(shape=(M,))
```

La línea de código anterior crea una capa de entrada y espera que los datos de entrada tengan una forma de $(M,)$, lo que significa que el modelo espera recibir datos unidimensionales con M elementos. Esta capa no suelen implementar funciones de activación ni de inicialización, debido a que la capa de entrada simplemente toma los datos de entrada y los envia a la siguiente capa, sin aplicar ninguna transformación no lineal.

4.2.3. Capa Densa

En una red neuronal una capa densa u oculta es la capa o conjunto de capas entre la capa de entrada y la capa de salida. Estas capas se encargan de realizar transformaciones no lineales en los datos de entrada, lo que permite a la red aprender representaciones más complejas y abstractas del conjunto de datos. Cada neurona de la capa oculta recibe la salida de la neurona de la capa anterior, realiza una operación de ponderación y aplica una función de activación.

Extracto de código 4.2: Implementación de capas densas para el codificador

```
encoded = Dense(M, activation='relu')(input_signal)
```

En esta línea de código se crea una capa densa o completamente conectada en el modelo:

- **Dense:** Es el nombre con la que Keras define una capa que implementa una operación de multiplicación de matriz entre la entrada y una matriz de pesos, seguida de una operación de sesgo (bias) y una función de activación.
- **M:** Es el número de neuronas en la capa densa.
- **activation='relu':** Especifica la función de activación utilizada después de la operación de multiplicación de matriz y sesgo. En este caso, se utiliza la función de activación Rectified Linear Unit (ReLU), que es común en muchas redes neuronales debido a su capacidad para introducir no linealidades en el modelo y superar problemas de desvanecimiento del gradiente.
- **(input_signal):** Conecta esta capa densa a la capa de anterior.

4.2.4. Capa de Normalización

La capa de normalización del transmisor es fundamental para garantizar que la señal de salida se ajuste a las limitaciones físicas de potencia o energía. Esto es importante porque evita que el auto codificador genere salidas demasiado grandes, que no sólo son innecesarias, sino que también pueden causar inestabilidad en el sistema. Esta inestabilidad se genera a valores de gradientes grandes que pueden provocar oscilaciones en el proceso de optimización, dificultando la convergencia y afectando el aprendizaje. Al imponer límites a la amplitud de salida, la capa de normalización garantiza que la potencia o energía de la señal permanezcan dentro de los parámetros requeridos. Además, controlar estas variables puede evitar el sobreajuste y mejorar la robustez del modelo.

4.2.4.1. Energía

En [6, 14, 38], la restricción de energía se expresa como:

$$\|x\|_2^2 \leq n \quad (4.1)$$

$$\left(\sqrt{\sum_{i=1}^n |x_i|^2} \right)^2 \leq n \quad (4.2)$$

Por lo tanto, la normalización para cada valor que ingresa se define como:

$$f(x) = \frac{x}{\sqrt{\|x\|_2^2}} = \frac{x}{\sqrt{\sum_{i=1}^n |x_i|^2}} \quad (4.3)$$

La implementación de una capa para normalización define como:

Extracto de código 4.3: Implementación de capa de normalización de energía

```
encoded2 = EnergyNormalizationLayer() (encoded1)
```

Donde:

- *EnergyNormalizationLayer()*: es una clase que define una capa personalizada.
- *encoded1*: es un tensor, un objeto matemático que pertenece a un espacio vectorial, que contiene los datos que ingresan a la capa de normalización.
- *encoded2*: es el tensor de salida

La implementación de la clase para la ecuación 4.3 se define como:

Extracto de código 4.4: Implementación de restriccion de energía

```
class EnergyNormalizationLayer( tf.keras.layers.Layer ):
    def __init__( self ):
        super( EnergyNormalizationLayer , self ) . __init__()
    def call( self , inputs ):
        n = inputs . shape [ 1 ] #longitud del vector de entrada
        s = tf . reduce_sum( tf . square( inputs ) , axis=1 , keepdims=True ) /
            ( n / 2 )
        normalized_inputs = inputs / tf . sqrt( s )
        return normalized_inputs
```

Este método de normalización de energía es una técnica que ajusta los datos para que tengan una energía comparable, lo que mejora la estabilidad del entrenamiento en redes neuronales.

4.2.4.2. Potencia promedio

En [6, 14, 38], la restricción de potencia media se expresa como:

$$\mathbb{E}[|x_i|^2] \leq 1, \quad \forall i \quad (4.4)$$

Por lo tanto, la capa de normalización se define como:

$$f(x) = \frac{\sqrt{Ex}}{\sqrt{\mathbb{E}[|x_i|^2]}} = \frac{\sqrt{Ex}}{\sqrt{\frac{1}{n} \sum_{i=1}^n |x_i|^2}} \quad (4.5)$$

La implementación de una capa para normalización se realiza de la siguiente manera:

Extracto de código 4.5: Implementación de capa de normalización de potencia promedio

```
encoded2 = PowerNormalizationLayer(n_channel)(encoded1)
```

Donde:

- *PowerNormalizationLayer()*: es una clase que define una capa personalizada.
- *n_channel*: número de neuronas a la salida de la capa.
- *encoded1*: es un tensor que contiene los datos que ingresan a la capa de normalización.
- *encoded2*: es el tensor de salida

La implementación de la clase para la operación de la ecuación 4.5 se define como:

Extracto de código 4.6: Implementación de restriccion de potencia promedio

```
class PowerNormalizationLayer(tf.keras.layers.Layer):  
    def __init__(self, n, epsilon=1e-5, momentum=0.99, **kwargs):  
        super(PowerNormalizationLayer, self).__init__(**kwargs)  
        self.epsilon = epsilon  
        self.momentum = momentum  
        self.n = n  
  
    def build(self, input_shape):  
        # Asegúrate de que input_shape es una tupla  
        input_shape = tf.TensorShape(input_shape)  
        if len(input_shape) == 1:  
            shape = (input_shape[-1],)  
        else:  
            shape = input_shape[-1:]  
  
        self.moving_mean = self.add_weight(name='moving_mean',  
                                           shape=shape,  
                                           initializer='zeros',  
                                           trainable=False)  
        self.moving_variance = self.add_weight(name='moving_variance',  
                                              shape=shape,  
                                              initializer='ones',
```

```

trainable=False)

def call(self, inputs, training=False):
    if training:
        batch_mean, batch_variance = tf.nn.moments(inputs, axes=[0])
        mean = batch_mean
        variance = batch_variance
        self.moving_mean.assign(self.moving_mean * self.momentum +
                               mean * (1 - self.momentum))
        self.moving_variance.assign(self.moving_variance * self.
                                     momentum + variance * (1 - self.momentum))
    else:
        mean = self.moving_mean
        variance = self.moving_variance

    normalized_inputs = ((inputs - mean) / tf.sqrt(2 * (variance +
                                                       self.epsilon)))
    E = tf.reduce_sum(tf.square(normalized_inputs), axis=1, keepdims
                      =True)
    P = tf.reduce_mean(E)
    return normalized_inputs

```

Este método de normalización de potencia promedio es una técnica que ajusta los datos para que tengan una potencia promedio comparable y acotada por un valor, alrededor de 1, lo que puede mejorar la estabilidad del entrenamiento en redes neuronales. Es importante notar que el código implementado es una versión más simple de la capa de *Batch Normalization* (librería de Keras) logrando un resultado similar a la salida de la capa. Se utilizó esta clase debido a que en esta se tiene un control absoluto sobre parámetros y comportamiento de la misma, lo cual no es posible a través del uso de la función que define la capa, provista por la librería.

4.3. Arquitectura del Canal

El modelo de un canal aditivo de ruido blanco Gaussiano, comúnmente conocido como Additive White Gaussian Noise (AWGN), agrega ruido al i vector que representa la señal. Este ruido,

denominado N , es un vector n-dimensional con elementos independientes e idénticamente distribuidos (i.i.d.), donde cada elemento tiene una distribución normal compleja con media 0 y varianza σ^2 , $N_i \sim CN(0, \sigma^2)$.

$$y(x) = x + w, \quad w \sim N(0, \sigma^2 I) \quad (4.6)$$

La capa del canal agrega ruido AWGN con una varianza

$$\sigma^2 = \frac{1}{2RkE_b/N_0} \quad (4.7)$$

Donde:

- **k**: es el número de bits por símbolo de la modulación a emplear.
- **E_b/N_0** : denota la relación entre la energía por bit (E_b) y la potencia de ruido, con densidad espectral N_0 .
- **R**: Es la tasa de código utilizada, que se define como: $R = k/n$ (*bits/uso de canal*), donde $k = \log_2 M$ es el número de bits se transmiten a través del uso de n canales, denotado por el par (n, k) .

La implementación de la capa de canal se define como:

Extracto de código 4.7: Implementación de capa de ruido WGN

```
EbNo_train_dB = 3
EbNo_train=10.0**(EbNo_train_dB / 10.0)
encoded3 = GaussianNoise(np.sqrt(1/(2*R*EbNo_train)))(encoded2)
```

Donde:

- **GaussianNoise**: es una capa en Keras que agrega ruido gaussiano a sus entradas durante el entrenamiento. Esto es útil para regularizar la red y hacerla más robusta al ruido en los datos de entrada.
- La expresión interior se usa para calcular la desviación estándar del ruido.

El ruido se agrega de la siguiente manera:

Extracto de código 4.8: Generación de ruido WGN

```
def call(self, inputs, training=None):  
    def noised():  
        w = self._random_generator.random_normal(  
            shape=tf.shape(inputs),  
            mean=0.0,  
            stddev=self.stddev,  
            dtype=inputs.dtype,  
        )  
        return inputs + w
```

Donde:

- *inputs*: Es el tensor de entrada que contiene los valores para agregar ruido.
- *self._random_generator.random_normal(...)* genera un tensor de números aleatorios con una distribución normal (Gaussiana).
 - *tf.shape(inputs)* obtiene la forma del tensor de entrada, *inputs*, de tal manera que el tensor de números aleatorios tenga la misma forma.
 - *mean=0.0*: el ruido tendrá una media centrada en cero.
 - *self.stddev* es un atributo de la clase que define la desviación estándar del ruido que se va a añadir. Este valor se envía como parámetro y determina la amplitud del ruido.
 - *inputs.dtype*: asegura que el tipo de datos del tensor de ruido sea el mismo que el del tensor de entrada *inputs*

4.4. Arquitectura del Decodificador

El receptor/decodificador del auto codificador, de acuerdo a la Figura 4.1, está formado por dos capas densas. La primera capa tiene neuronas con una activación ReLU, y la segunda capa tiene neuronas con una activación SoftMax, que se utiliza para crear el vector de probabilidad el cual ayuda a la estimación del símbolo final.

4.4.1. Capa Densa 1

En el decodificador, esta primera capa cuenta con M neuronas y una función de activación ReLU. En este contexto, cuando las capas densas en una red neuronal se conectan, cada neurona de la capa de salida está conectada a cada neurona en la siguiente capa, dicha conexión está asociada a los pesos que la red neuronal aprende durante el entrenamiento.

Extracto de código 4.9: Implementación de capa densas para el decodificador

```
decoded = Dense(M, activation='relu')(encoded3)
```

Esta línea de código crea una capa densa completamente conectada en el modelo:

- **Dense:** Capa en Keras que implementa una operación teniendo en consideración el número de neuronas, M , y el número de datos a la entrada de la capa, n , donde n corresponde al número de canales. Esto significa que cada una de las M neuronas está conectada a las n neuronas de la entrada obteniendo una matriz de $M \times n$ pesos que conectan estas capas. Para los casos en que $n < M$ la red realiza una representación latente de n a M dimensiones, la cual se consigue mediante la multiplicación de la representación de entrada con n dimensiones por la matriz de pesos con $M \times n$ dimensiones seguida de una adición de un sesgo (bias) y el resultado es el argumento de la función de activación.
- **M :** número de neuronas en la capa densa, y por ende el número de datos de entrada con los que se cuenta para la capa siguiente.
- **activation='relu'**: Función de activación implementada, la cual se utiliza con el objetivo de introducir la no linealidad en los datos. Al aplicar ReLU se activan las neuronas cuyas salidas son positivas, mientras que las neuronas con salidas negativas se desactivan (tienen una salida de cero).
- **encoded3**: Indica la capa anterior (capa de canal) que se conecta con la capa actual.

4.4.2. Capa Densa 2

La línea de código a continuación crea una capa densa completamente conectada en el modelo:

Extracto de código 4.10: Implementación de segunda capa de densa decodificador

```
decoded1 = Dense(M, activation='softmax')(decoded)
```

- **activation='softmax'**: Función de activación implementada, la cual se utiliza en los problemas de clasificación multiclase como es el caso del auto codificador . Al aplicar Softmax se produce a la salida un vector de probabilidades de tamaño M , donde cada elemento del vector se relaciona con la probabilidad de que la entrada recibida pertenezca a una de las M clases posibles. Entre las M probabilidades, que suman 1, se elige la de mayor valor. El índice correspondiente en el vector indica a cuál de los M valores pertenece la entrada.
- **decoded**: Indica la capa anterior (capa densa 1 del decodificador) que se conecta con la capa actual.

4.4.3. Datos de Salida

Para determinar a que clase pertenece los datos de salida de la red se utiliza en la parte de inferencia de la red la función argmax la cual se encuentra implementada en la siguiente línea de código:

Extracto de código 4.11: Generar datos de salida

```
pred_output = np.argmax(pred_final_signal, axis=1)
```

- **pred_final_signal**: Contiene en cada de unas sus columnas las probabilidades obtenidas a la salida de la capa densa 2 del decodificador para cada una de las clases (símbolos) introducida en el auto codificador.
- **argmax, axis=1**: Indica que la búsqueda se realiza a lo largo de cada fila y se selecciona el índice de la columna con el valor más alto, es decir, la clase con la probabilidad más alta.
- **pred_output**: Finalmente, los índices escogidos se almacenan en un vector; esto servirá para posteriormente realizar una comparación con la entrada original y la salida predicha.

4.5. Entrenamiento de la Red

4.5.1. Generar modelo

Una vez definida las capas del auto codificador, se crea el modelo utilizando la API funcional de Keras, como se indica en las siguientes líneas de código. El parámetro *input_signal* es el tensor de entrada del modelo y *decoded1* es el tensor de salida del modelo, que representa la reconstrucción del *input_signal* después de pasar por el proceso de codificación y decodificación.

Extracto de código 4.12: Generar modelo final del auto codificador

```
#crear modelo
autoencoder = Model(input_signal , decoded1)
#Configurar modelo
autoencoder.compile(optimizer='adam' , loss=
                     categorical_crossentropy , metrics=['accuracy'])
```

La compilación configura el modelo para su entrenamiento, donde:

- **optimizer= ‘adam’** especifica que se usará el optimizador Adam para ajustar los pesos del modelo durante el entrenamiento.
- **loss=‘categorical_crossentropy’** establece la función de pérdida que se usará para evaluar el error durante el entrenamiento, la cual es adecuada para problemas de clasificación categórica.
- **metrics= ‘accuracy’** indica que la métrica de precisión se utilizará para evaluar el rendimiento del modelo durante el entrenamiento y la validación.

Antes de proceder con el entrenamiento de la red, es necesario realizar las configuraciones generales que se indican en la Tabla 4.3.

Para la etapa de entrenamiento, el auto codificador utiliza el algoritmo de retropropagación, el cual es considerado fundamental para entrenar redes neuronales de este tipo. Este algoritmo ajusta iterativamente los pesos de la red neuronal para minimizar la función de pérdida en los datos de entrenamiento. *EarlyStopping* monitorea la métrica de pérdida (*loss*) y detiene el entrenamiento si no se obtiene una mejora después de un número específico de Épocas *epoch*, determinado por *patience*, indica el número de épocas que esperará el algoritmo

Tabla 4.3: Parámetros de Entrenamiento

Parámetros de Entrenamiento	Descripción
Datos de Entrenamiento	1000000
Datos de Validación	100000
Número de Épocas	50
Tamaño de Lote	1024
Función de Pérdida	Entropía Cruzada Categórica
Algoritmo de Optimización	Adam
EarlyStopping	5
Tasa de Aprendizaje	0,001

para detener el entrenamiento luego de no obtener una mejora. Además, el parámetro `restore_best_weights=True` asegura que los pesos del modelo se restauren al estado en el que se obtuvo la mejor métrica de pérdida.

Extracto de código 4.13: Implementación del entrenamiento

```
#early_stopping
early_stopping = EarlyStopping(monitor='loss', patience=5,
                               restore_best_weights=True)

#Entrenamiento
historial = autoencoder.fit(data, data,
                             epochs=50,
                             batch_size=1024,
                             validation_data=(val_data, val_data),
                             callbacks=[early_stopping])
```

Luego de generar y configurar el modelo se procede a entrenar el auto codificador, donde:

- **data:** son los datos de entrada y también los datos de salida deseados.
- **epochs:** establece el número máximo de épocas para el entrenamiento, 50.
- **batch_size:** define el tamaño del lote en 1024, que es el número de muestras que se

procesan antes de actualizar los pesos del modelo.

- **validation_data:** proporciona los datos de validación para evaluar el modelo en cada época.
- **callbacks:** incluye el callback *early_stopping* definido anteriormente para monitorear y controlar el proceso de entrenamiento.

5. Análisis y Resultados

En este Capítulo se muestran los resultados obtenidos con la implementación del AE en diversos escenarios, es decir con diferentes esquemas de modulación. En las Secciones siguientes, se presentan los resultados a través de las gráficas correspondientes a las métricas de función de pérdida y función de validación durante el entrenamiento del AE. Así mismo, se incluyen las representaciones gráficas de los diagramas de constelación para los distintos esquemas, tanto en la salida del codificador como en la salida del canal, con el fin de observar el impacto del ruido AWGN en la interpretación de la calidad de la señal. Para cada escenario, se muestra el rendimiento obtenido en términos del Block Error Rate (BLER) y Bit Error Rate (BER), los cuales han sido evaluados para varios niveles de SNR en el rango de 0 – 14 dB, así como para diversas tasas de codificación. En la Sección 5.12, se presenta un resumen del análisis de los resultados obtenidos para cada parámetro, con el propósito de aclarar los beneficios y los desafíos vinculados a la implementación de los auto codificadores.

5.1. Métricas de evaluación para el desempeño del auto codificador

En el contexto de la clasificación de datos multiclase con redes neuronales, tanto las métricas de evaluación como las de validación cumplen roles importantes en la optimización y valoración del modelo. Las métricas de evaluación, como la precisión (en inglés, accuracy), la precisión por clase (en inglés, precision), la exhaustividad por clase (en inglés, recall), la puntuación F1 por clase y la matriz de confusión extendida, proporcionan una comprensión detallada de cómo el modelo maneja la clasificación entre múltiples categorías. Estas métricas ayudan a identificar las fortalezas y debilidades del modelo, orientando los ajustes necesarios.

Las métricas de evaluación, como la función de pérdida o costo, cuantifica el error los valores reales y los valores de predicción del modelo durante el entrenamiento. Al minimizar esta pérdida, el modelo ajusta sus pesos para mejorar su capacidad de diferenciación entre clases. Para el modelo implementado se evaluará la función de pérdida para la optimización del modelo y la precisión para estimar la proporción de predicciones correctas sobre el total de predicciones.

En la etapa de predicción se evaluará el BLER y el BER que son métricas que proporcionan una medida cuantitativa de la calidad de la transmisión de datos al usar un auto codificador. Para garantizar la confiabilidad de los resultados, se calculará la media de 10 simulaciones y se calcula el intervalo de confianza del 95 %.

5.2. BPSK - auto codificador (2,1)

5.2.1. Entrenamiento

En la figura 5.1 se muestra los resultados del entrenamiento del auto codificador empleando $n = 2$ canales y $k = 1$ bit/símbolo. La gráfica de la izquierda corresponde a la función de pérdida, donde:

- Eje Y (Pérdida): Representa el valor de la función de pérdida, que mide la discrepancia entre las predicciones del modelo y los valores reales. Un valor más bajo indica un mejor ajuste del modelo a los datos de entrenamiento.
- Eje X (Épocas, en inglés Epoch): Indica el número de épocas, o ciclos completos de entrenamiento en los que el modelo ha sido ajustado usando todo el conjunto de datos de entrenamiento.
- Valor de pérdida: Indica los valores de la función de pérdida en cada época durante el entrenamiento. Representado por puntos azules.
- Valor de validación: Indica los valores de la función de pérdida para el conjunto de validación en cada época. Representado por una línea azul.

En la Figura 5.1, la gráfica de la derecha corresponde a la función de precisión, donde:

- Eje Y (Precisión): Representa la proporción de predicciones correctas realizadas por el modelo. Un valor de 1.0 indica una precisión del 100 %.
- Eje X (Épocas, en inglés Epoch): Similar que en el caso de la función de pérdida, muestra el número de épocas de entrenamiento.
- Valor de precisión: Indica los valores de precisión en cada época durante el entrenamiento. Representado por puntos rojos
- Valor de validación: Indica los valores de precisión para el conjunto de validación en cada época. Representado por una línea roja.

Al inicio del entrenamiento, la pérdida es relativamente alta, lo que es común antes que el modelo comience a aprender de los datos. Si se toma en cuenta que la pérdida se calcula para cada muestra y luego se promedia, considerando que al inicio todas las predicciones serán iguales, el valor promedio aproximado de la función de pérdida se puede obtener al

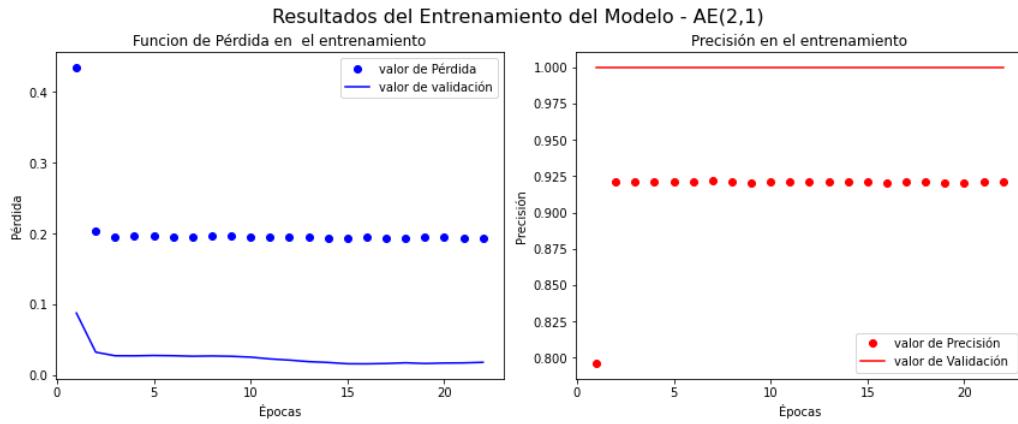


Figura 5.1: Resultados del entrenamiento para el auto codificador (2,1) normalizado en energía.

emplear la ecuación 2.20 para el caso de Binary Phase Shift Keying (BPSK):

$$L = - \sum_{i=1}^N (y_{ij} \ln(p_{ij})) \quad (5.1)$$

Donde:

- M es el número de clases.
- y_i es el vector one-hot de la clase verdadera, que puede ser 0 o 1.
- p_i es la probabilidad estimada de que la muestra i pertenezca a la clase posible.

$$L = - (y_{1j} \ln(p_{1j}) + (1 - y_{1j}) \ln(1 - p_{1j})) \quad (5.2)$$

Dado que y_i es una representación one-hot, solo uno de las entradas y_{ij} será 1 y el otro será 0, esto ocurre de manera similar para las demás modulaciones. Por lo tanto:

$$L = - \ln(0.5) \approx 0.693 \quad (5.3)$$

Al observar la gráfica izquierda del entrenamiento en la figura 5.1 el valor de pérdida es de ≈ 0.45 , que difiere del calculado, debido a las particularidades de la inicialización de pesos que en un inicio son inicializados aleatoriamente. Con el progreso de las épocas, la pérdida disminuye rápidamente en las primeras y se estabiliza alrededor de ≈ 0.2 , debido a la distribución de los datos; este comportamiento muestra una mejora en el ajuste del modelo a los datos de entrenamiento. Por otro lado, la línea continua azul muestra que los valores de pérdida en el conjunto de validación siguen una tendencia similar, con la diferencia que los valores de pérdida son inferiores ≈ 0.02 , lo cual es un buen indicador de que el modelo está generalizando adecuadamente y no está sobre ajustándose a los datos de entrenamiento.

En cuanto a la precisión, al inicio es baja ≈ 0.82 , lo que es esperado antes de que el modelo comience a aprender de los datos. A medida que avanzan las épocas, la precisión en el entrenamiento se estabiliza en un valor alto ≈ 0.925 , indicando que el modelo está haciendo predicciones correctas con mayor frecuencia. Sin embargo, la línea continua roja en el gráfico muestra que la precisión en el conjunto de validación también se mantiene alta y estable ≈ 1 , sugiriendo que el modelo está generalizando bien datos no vistos durante el entrenamiento. Finalmente, en la tabla 5.8 y a manera de resumen se presentan todos los escenarios que se van a implementar y son fundamentales en el entrenamiento para este auto codificador (2,1).

5.2.2. Diagrama de Constelación

De acuerdo a [38] en un sistema de comunicaciones basado en auto codificadores, los diagramas de constelación de salida no están predefinidos sino que se aprenden en función de la métrica de rendimiento a optimizar. Con el objetivo de demostrar lo antes citado, se procede a graficar las constelaciones para las diferentes tasas empleadas.

Tabla 5.1: Valores del auto codificador (2,1) normalizado en energía.

S1	-0.824752	-0.5654931	S2	0.82582	0.56393
-----------	-----------	------------	-----------	---------	---------

En la tabla 5.1 están los valores para cada símbolo, a la salida del codificador. En la figura 5.2 se observan 3 diagramas de constelación para el auto codificador (2,1); estos diagramas son muy similares al de BPSK convencional. La imagen de la izquierda muestra el diagrama de constelación sin la presencia de ruido. En este caso, las señales moduladas se encuentran exactamente en sus posiciones ideales en el plano complejo, distribuidas uniformemente en el círculo unitario.

En la imagen del centro, el diagrama de constelación muestra la misma señal modulada pero con un nivel de ruido añadido de 10 dB. El ruido introduce perturbaciones en la posición de cada punto, lo que causa que los puntos se dispersen alrededor de sus posiciones ideales. Esta presencia de ruido incrementará la probabilidad de error en la demodulación, ya que algunos puntos pueden superponerse o estar muy cerca entre sí.

La imagen de la derecha, corresponde al diagrama de constelación con un SNR de 20 dB. En este caso, el nivel de ruido es menor en comparación con el de 10 dB; resultando en una menor dispersión de los puntos alrededor de sus posiciones ideales, permitiendo que la demodulación

sea más precisa y en consecuencia, una menor probabilidad de error que en el caso anterior.

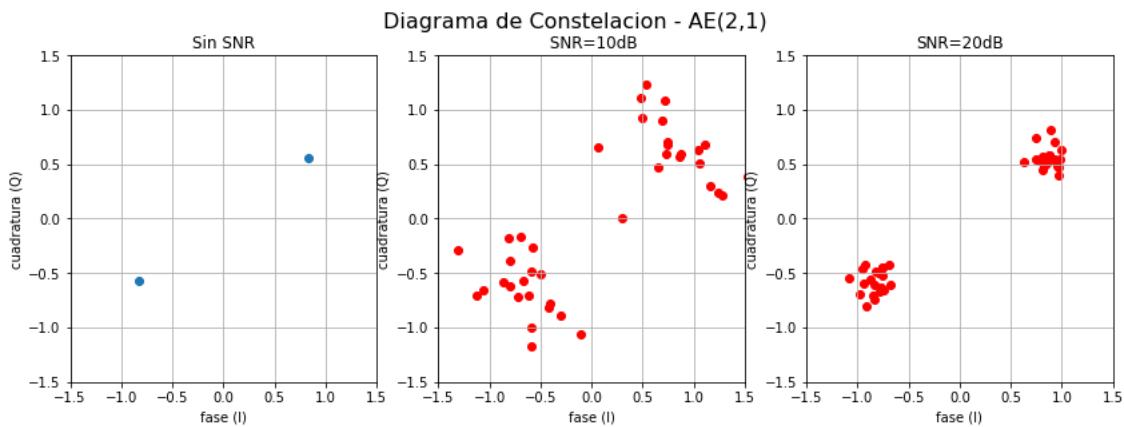


Figura 5.2: Diagrama de constelación para el auto codificador (2,1) normalizado en energía.

5.2.3. Probabilidad de Error

La gráfica de probabilidad de error de bloque (BLER) es una herramienta importante en el análisis y diseño de sistemas de comunicación ya que facilita la evaluación del algoritmo de codificación y modulación. Para obtener estas gráficas de BLER se procedió a realizar 10 simulaciones, con un E_b/N_0 de 0 a 14 dB, con el objetivo de obtener una media representativa de los datos y el intervalo de confianza del 95 %. En la tabla 5.2 se observan los valores obtenidos para el auto codificador (2,1).

En la Figura 5.3 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (2,1) y la modulación BPSK. La gráfica izquierda corresponde a una escala lineal, donde se aprecia que la línea azul, representa al BLER, mismo que disminuye a medida que aumenta la relación de E_b/N_0 . En cuanto a los intervalos de confianza se puede observar que son pequeños; por lo tanto, la dispersión entre los datos de las diferentes simulaciones es mínima lo que sugiere una alta precisión y similitud en las mediciones.

En la gráfica de la derecha se presenta el BLER en escala logarítmica. Se puede notar que el BLER para el auto codificador, línea azul, muestra un rendimiento inferior en comparación con el teórico correspondiente para BPSK. Este rendimiento se debe a factores como la eficacia del proceso de optimización, es posible que el AE se encuentre sobreajustado a los datos de entrenamiento ya que para el set de entrenamiento únicamente se dispone de 2 símbolos, que al ser representados en una codificación *one-hot* la variabilidad que puede tener el set es muy baja; por lo tanto, la capacidad de generalización del sistema también es baja. Una solución sería reducir el tamaño del set de entrenamiento o incluir más redundancia.

Tabla 5.2: Valores de BLER con su respectivo IC para el auto codificador (2,1) normalizado en energía.

E_b/N_0 (dB)	BLER media	IC 95 %
0	0,1584169	$\pm 2,0049E-04$
2	0,1042007	$\pm 2,5565E-04$
4	0,0564626	$\pm 1,9018E-04$
6	0,0229916	$\pm 1,7026E-04$
8	0,0059991	$\pm 5,5290E-05$
10	0,0007899	$\pm 1,9990E-05$
12	3,35E-05	$\pm 4,8687E-06$
14	2,00E-07	$\pm 3,4400E-07$

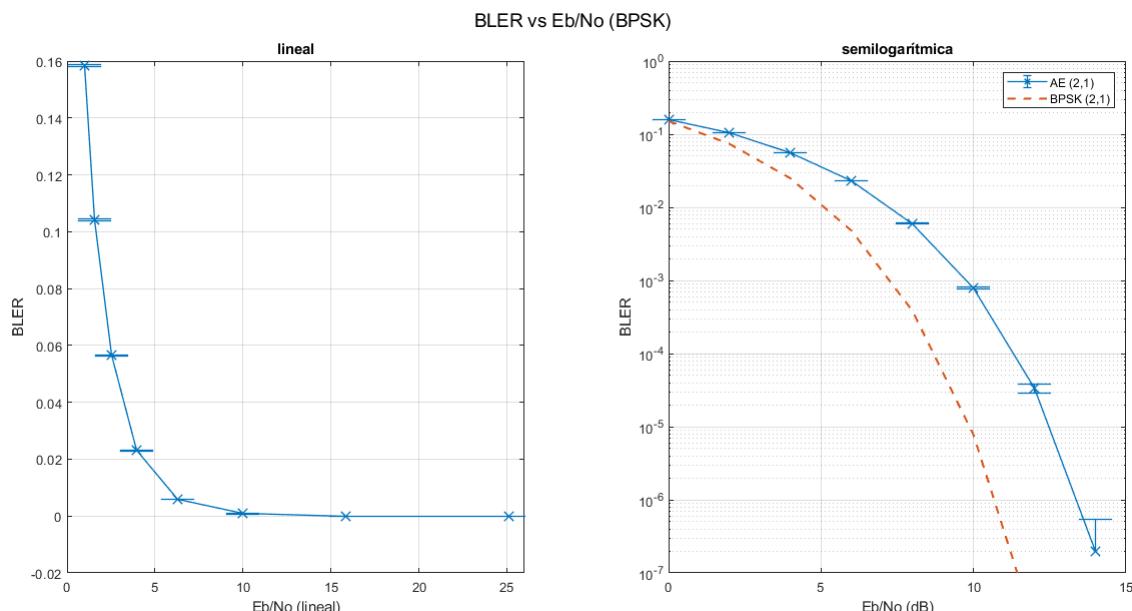


Figura 5.3: Gráfica del BLER para auto codificador (2,1) normalizado en energía.

Si bien el BLER proporciona una evaluación amplia y útil para analizar la efectividad de las técnicas de codificación y corrección de errores en bloques completos de datos, otra métrica importante es el BER. El BER mide la precisión con la que los bits individuales se transmiten desde el emisor al receptor, permitiendo evaluar el rendimiento de la transmisión de datos a

nivel de bits. El BER en términos de BLER se puede obtener mediante la ecuación:

$$BER = 1 - (1 - BLER)^{\frac{1}{n}} \quad (5.4)$$

Donde:

- n: es el tamaño del bloque de bits mas la redundancia, a la salida del codificador para la transmisión.

En la tabla 5.3 se observa los valores de BER, obtenidos a partir de la ecuación 5.4, y su intervalo de confianza del 95 % para este mismo auto codificador. En la figura 5.4 se observa la probabilidad de error de bit obtenida. La gráfica de la izquierda muestra el BER en escala lineal; mientras que en la gráfica de la derecha, en escala logarítmica. Tanto la gráfica del AE como la de BPSK no codificado comienzan con el mismo valor de BER en 0 dB ; sin embargo, el rendimiento observado para el AE es inferior, en aproximadamente 2 dB, que el de BPSK. Esto confirma que el modelo de AE tampoco es capaz de decodificar correctamente bit a bit, a pesar de tener un buen entrenamiento; el rendimiento no ha sido el esperado en la inferencia u operación.

Tabla 5.3: Valores de BER con su respectivo IC para el auto codificador (2,1) normalizado en energía.

E_b/N_0	BER	IC 95 %
0	0,08262162	$\pm 8,30E-05$
2	0,05353327	$\pm 1,03E-04$
4	0,02864147	$\pm 7,44E-05$
6	0,01156265	$\pm 6,54E-05$
8	0,00300406	$\pm 2,11E-05$
10	0,00039503	$\pm 7,60E-06$
12	1,68E-05	$\pm 1,85E-06$
14	1,00E-07	$\pm 1,31E-07$

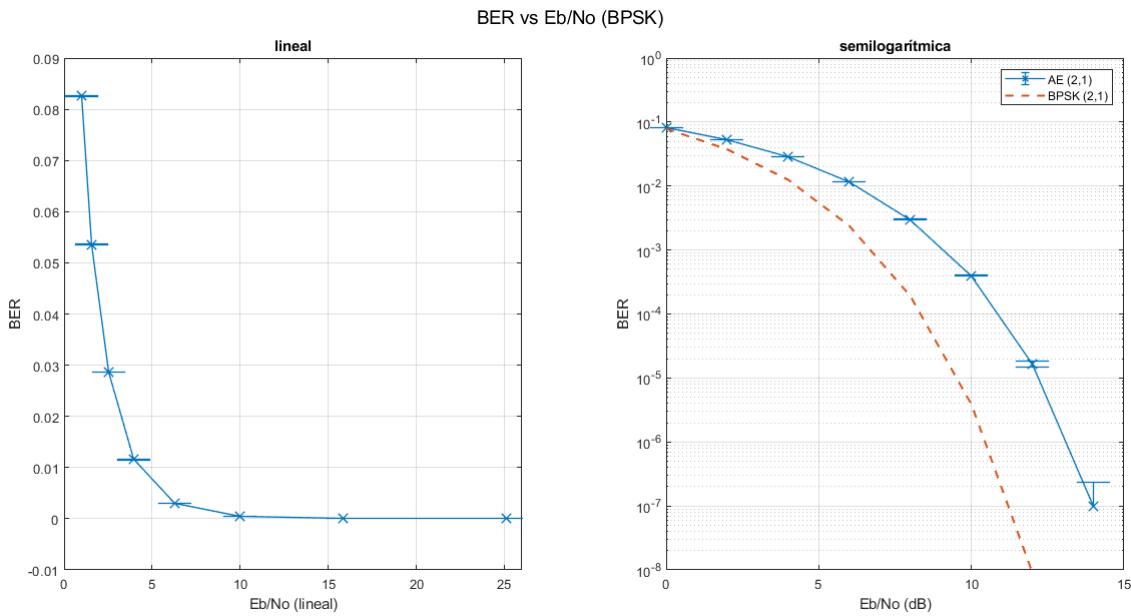


Figura 5.4: Gráfica del BER para el auto codificador (2,1) normalizado en energía.

5.3. QPSK - auto codificador (n,2)

5.3.1. Entrenamiento

La primera configuración para el auto codificador es analizada (2,2), que emplea $n = 2$ canales y $k = 2$ bits/símbolo, produciendo una tasa $R = 1$, sin redundancia. En la tabla 5.8 se muestra a manera de resumen el tiempo de entrenamiento total para esta tasa de código. En la Figura 5.5 se muestra los resultados del entrenamiento de este AE (2,2).

De manera similar al entrenamiento de BPSK, el valor de pérdida calculado teniendo en cuenta $M = 4$ símbolos o clases con probabilidad de $p_{ij} = 0.25$:

$$L = - \sum_{i=1}^4 (y_{ij} \ln(0.25)) = 1.3862 \quad (5.5)$$

El valor de pérdida inicial observado es de ≈ 1 , el cual decrece hasta ≈ 0.18 para el entrenamiento; mientras que para el set de validación alcanza ≈ 0 luego 25 épocas. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.65 y alcanza ≈ 0.95 ; sin embargo, para el set de validación alcanza ≈ 1 sugiriendo que el modelo está generalizando adecuadamente los datos diferentes a los de entrenamiento.

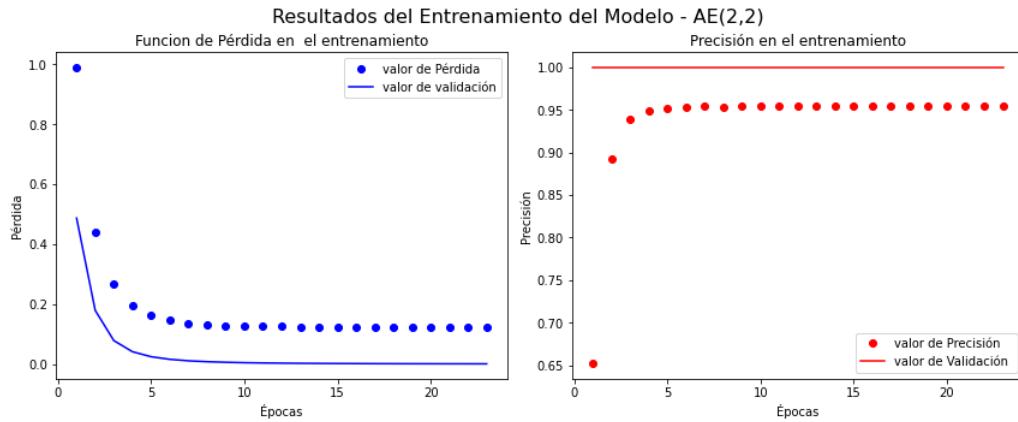


Figura 5.5: Resultados del entrenamiento para el auto codificador (2,2) normalizado en energía.

El segundo caso indicado en la Figura 5.6, es para el auto codificador (3,2) que produce una tasa $R = \frac{2}{3}$; es decir, se agrega 33 % de redundancia. El valor de pérdida inicial observado, en el entrenamiento, es de ≈ 0.9 , el cual decrece hasta ≈ 0.1 para el entrenamiento; mientras que para el set de validación alcanza ≈ 0 luego de 27 épocas. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.73 y alcanza ≈ 0.96 en el entrenamiento; sin embargo, para el set de validación alcanza ≈ 1 .

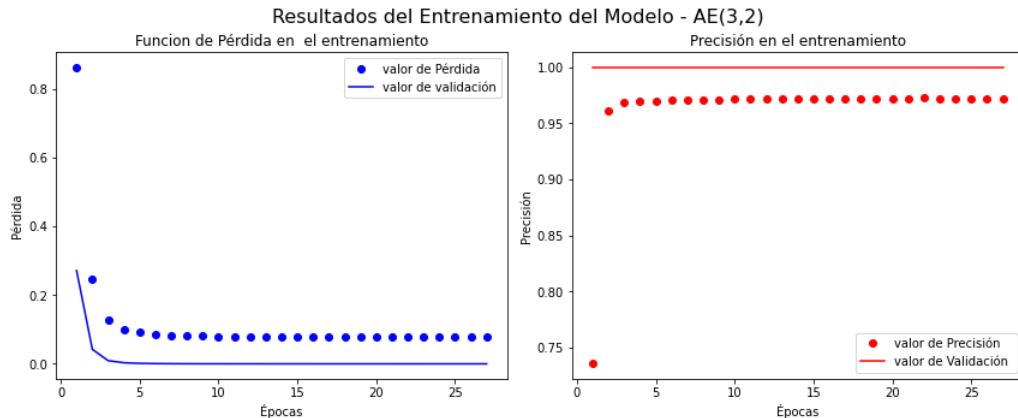


Figura 5.6: Resultados del entrenamiento para el auto codificador (3,2) normalizado en energía.

El tercer caso, presentado en la Figura 5.7, corresponde al auto codificador (4,2), que produce una tasa $R = \frac{1}{2}$; es decir, se agrega 50 % de redundancia. En la tabla 5.8 se puede observar el tiempo de entrenamiento total para esta tasa de código. El valor de pérdida calculado también es de ≈ 1.3862 , pero para este caso el valor observado es de ≈ 0.7 y se estabiliza alrededor de ≈ 0.2 para el entrenamiento y ≈ 0 en la validación, alcanzando las 27 épocas; estos valores son similares a los analizados en la figura 5.5 para una tasa $R = 1$ pero con 2 épocas menos. En cuanto a la gráfica de la precisión se observa que alcanza ≈ 0.97 en el entrenamiento y

≈ 1 para la validación. La precisión en el entrenamiento aumentó por $\approx 2.7\%$, debido al uso de 2 canales, pero la precisión en la validación es similar en los dos casos. En conclusión, el entrenamiento de estos 2 modelos ha permitido alcanzar un rendimiento similar para la clasificación.

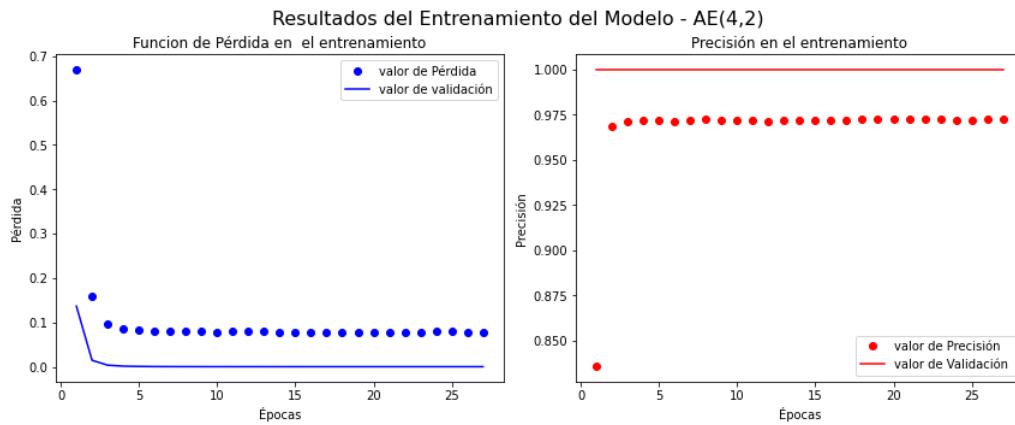


Figura 5.7: Resultados del entrenamiento para el auto codificador(4,2) normalizado en energía.

En definitiva, los mejores resultados de entrenamiento se obtuvieron con las tasas de código $R = 1$, $R = \frac{2}{3}$ y $R = \frac{1}{2}$. Estas dos últimas tasas permiten agregar bits de redundancia a la información, lo cual mejora la corrección de errores. Sin embargo, esto implica el aumento en la cantidad de canales, por ende un mayor número de épocas para un entrenamiento óptimo, y en consecuencia un mayor tiempo y recursos de procesamiento.

5.3.2. Diagrama de Constelación

En la tabla 5.4 se muestran los valores de salida del codificador. Mientras que en la figura 5.8 se observa que el diagrama de constelación del auto codificador (2,2) converge a el de una modulación Quadrature Phase Shift Keying (QPSK) convencional. La figura de la izquierda muestra un diagrama sin la presencia de ruido; para este caso, las señales moduladas se encuentran exactamente en sus posiciones ideales en el plano complejo y distribuidas uniformemente en el círculo unitario.

En la figura del centro, el diagrama muestra la misma señal modulada de la izquierda, pero con un nivel de E_b/N_0 de 10 dB, el cual introduce perturbaciones en la posición de cada punto causando que los puntos se dispersen alrededor de sus posiciones ideales. La Figura de la derecha muestra la constelación con un SNR de 20 dB, el cual al tener el nivel de ruido menor que en el caso anterior, la dispersión de los puntos también disminuye.

Tabla 5.4: Valores del auto codificador (2,2) normalizado en energía.

S1	0.02497	0.9996	S3	-0.9993	0.0353
S2	-0.0344	-0.9994	S4	0.9998	-0.0172

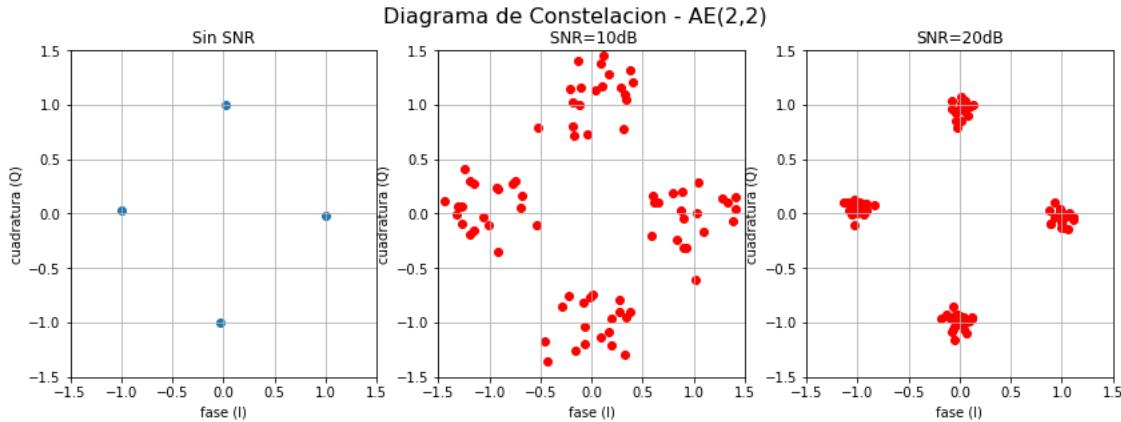


Figura 5.8: Diagrama de constelación para el auto codificador (2,2) normalizado en energía.

Tabla 5.5: Valores del auto codificador (4,2) normalizado en energía.

S1	0.87201	-0.747358	-0.804938	-0.182012
S2	-0.714062	0.832353	-0.812441	0.370464
S3	0.360865	0.667347	0.777697	-0.905324
S4	-0.506055	-0.746347	0.833525	0.701506

Al usar 2 canales cada símbolo se representa con dos valores, que directamente se pueden presentar en un diagrama de fase y cuadratura. Al usar más canales, para agregar redundancia, es necesario realizar una reducción de dimensionalidad, o proyección, a dos dimensiones para poder representar cada símbolo. En la figura 5.9 se observa la reducción de dimensión 3 a 2 para el Autoencoder (AE) (3,2). La gráfica de la izquierda, empleado al método de T-distributed Stochastic Neighbor Embedding. (t-SNE), muestra una distribución de los símbolos que no se asemeja a una constelación QPSK, pero trata de mantener la distancia con los otros símbolos. La gráfica de la derecha usa el método de Uniform Manifold Approximation and Projection (UMAP) y su resultado es una distribución que se asemeja más a las constelaciones ya conocidas. En la tabla 5.5 se pueden observar los valores generados por el auto codificador. En la figura 5.10 izquierda se aplica el algoritmo de t-SNE en Python para reducir la dimensionalidad.

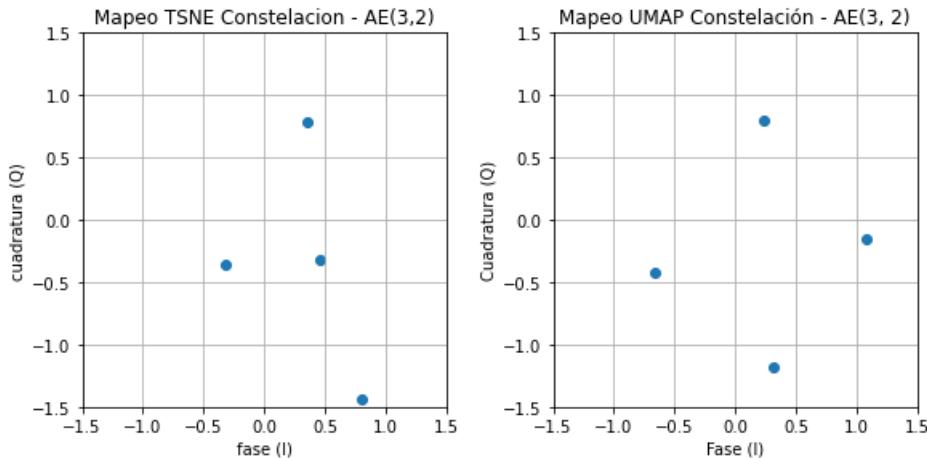


Figura 5.9: Constelaciones proyectadas para el auto codificador (3,2) normalizado en energía.

dad, donde se puede ver que dos símbolos se encuentran en un radio interno y dos símbolos en un radio externo. En la figura 5.10 derecha se observa la proyección empleando el algoritmo UMAP, el cual se parece más a la constelación observada en la figura 5.8, QPSK. De esta manera, se puede indicar que la proyección observada depende del algoritmo empleado, la cual resulta de utilidad debido a que nos permite tener una idea de como se distribuyen los símbolos en el espacio n dimensional. ¹²

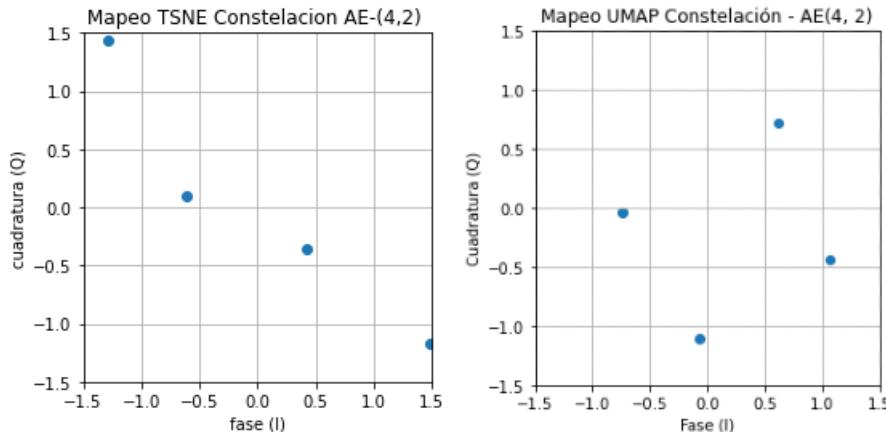


Figura 5.10: Constelaciones proyectadas para el auto codificador (4,2) normalizado en energía.

5.3.3. Probabilidad de Error

En la tabla 5.6 se observan los valores de la media para el BLER y su respectivo intervalo de confianza para el auto codificador (n,2). En la figura 5.11 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0). La gráfica izquierda corresponde a una escala lineal, donde se aprecia que las curvas del BLER disminuyen a medida que aumenta el E_b/N_0 . Al

Tabla 5.6: Valores de BLER con su respectivo IC para el auto codificador (n,2) normalizado en energía.

E_b/N_0 (dB)	(2,2)		(3,2)		(4,2)	
	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %
0	0,1512957	$\pm 2,9451E-04$	0,1213644	$\pm 2,1867E-04$	0,1211126	$\pm 1,62E-04$
2	0,0736592	$\pm 1,6253E-04$	0,0509307	$\pm 1,6497E-04$	0,0505991	$\pm 1,04E-04$
4	0,0248612	$\pm 8,3654E-05$	0,0132284	$\pm 7,6432E-05$	0,0131068	$\pm 9,14E-05$
6	0,0047838	$\pm 4,6194E-05$	0,0016379	$\pm 1,8966E-05$	0,001627	$\pm 3,08E-05$
8	0,0003879	$\pm 1,3483E-05$	6,19E-05	$\pm 5,4384E-06$	6,20E-05	$\pm 4,08E-06$
10	3,88E-04	$\pm 1,6476E-06$	1,00E-07	$\pm 1,9600E-07$	6,00E+00	$\pm 4,33E-07$
12	1,00E-07	$\pm 1,9600E-07$	0	0	0	0
14	0	0	0	0	0	0

variar la tasa de código a $R = \frac{2}{3}$ (AE(3,2)) y $R = 1/2$ (AE(4,2)) se puede observar que tiene un rendimiento similar. Los intervalos de confianza son pequeños; por lo tanto, la dispersión entre los datos es pequeño lo que sugiere una alta precisión y similitud entre estos.

En la gráfica derecha corresponde al BLER pero en escala logarítmica. Aquí se compara el rendimiento de un auto codificador (n,2) frente a un sistema QPSK no codificado. La curva para el AE, curva azul, muestra un rendimiento similar con la curva teórica (QPSK(2,2)), es decir, que el AE sin codificación, tasa $R = 1$, es capaz de recuperar los datos transmitidos con una misma probabilidad de error de bloque que para QPSK sin codificación.

Al aumentar el valor n , la tasa de código R se reduce permitiendo agregar redundancia para la detección y corrección de errores. En la gráfica se puede observar que con tasa $R = \frac{2}{3}$ (AE(3,2)), con un bit de redundancia, y la tasa $R = \frac{1}{2}$ (AE(4,2)), con 2 bits de redundancia, alcanzan un rendimiento similar hasta los 8 dB. Luego de este valor la tasa $R = \frac{1}{2}$ tiene un ligero incremento debido a que al tener dos bits de redundancia, el sistema posiblemente no es capaz de identificar y corregir el error.

Para analizar en términos de BER, en la tabla 5.7 se observan los valores de BER con sus respectivos intervalos de confianza; mientras que en la figura 5.12 se muestran las curvas en

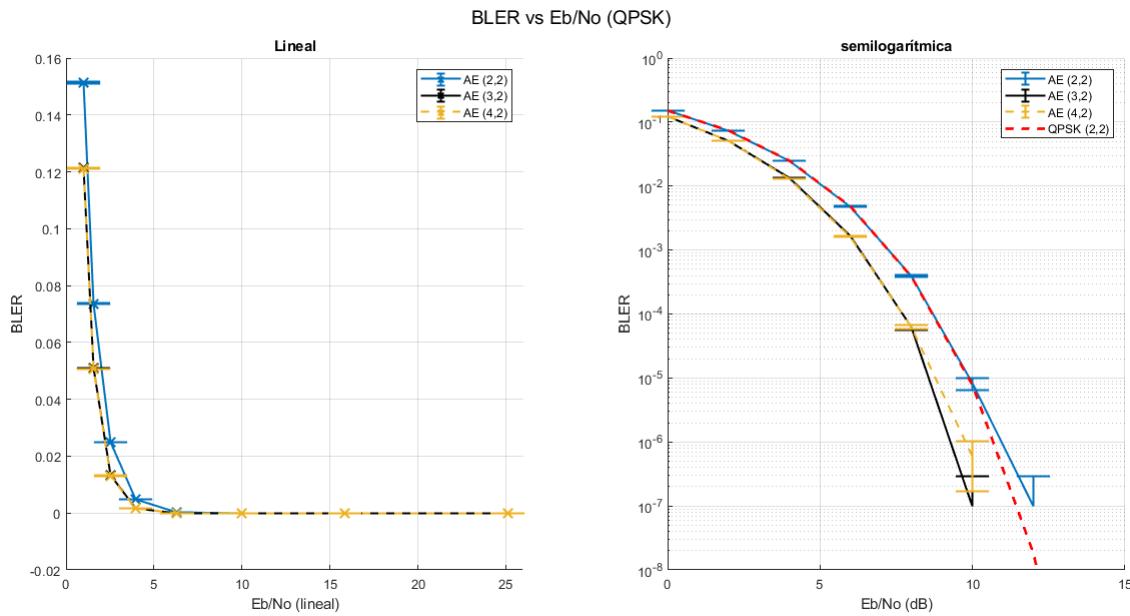


Figura 5.11: Gráfica del BLER para auto codificador (n,2) normalizado en energía.

escala lineal y logarítmica. Tanto el auto codificador (2,2) como el esquema de QPSK sin codificación comienzan en un mismo valor de BER, indicando que estos dos tienen un rendimiento similar. Para los otros esquemas de auto codificadores, también se observa un mejor rendimiento que QPSK, siendo el auto codificador (4,2) el de mejor rendimiento hasta un Eb/No de 8 dB. Al comparar con la gráfica del BLER, presentado en la Figura 5.11, es notable que tiene un comportamiento similar; sin embargo, las curvas de BER empiezan en un valor más pequeño indicando que la probabilidad de error a nivel de bit es menor, por debajo de 10^{-1} . Si el BER es menor al BLER se puede deducir que aunque hay pocos bits erróneos en general, esos errores están distribuidos de manera que afectan a muchos bloques y en consecuencia el BLER aumenta.

Tabla 5.7: Valores de BER con su respectivo IC para el auto codificador ($n,2$) normalizado en energía.

E_b/N_0	(2,2)		(3,2)		(4,2)	
	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %
0	0,07874855	$\pm 0,00015981$	0,04221156	$\pm 7,95E-05$	0,03175936	$\pm 4,46E-05$
2	0,03753401	$\pm 8,44E-05$	0,01727357	$\pm 5,69E-05$	0,01289714	$\pm 2,71E-05$
4	0,01250884	$\pm 4,24E-05$	0,00442906	$\pm 2,57E-05$	0,00329293	$\pm 2,31E-05$
6	0,00239477	$\pm 2,32E-05$	0,00054627	$\pm 6,33E-06$	0,000407	$\pm 7,71E-06$
8	0,00019397	$\pm 6,74E-06$	2,06E-05	$\pm 1,81E-06$	1,55E-05	$\pm 1,02E-06$
10	4,10E-06	$\pm 8,24E-07$	3,33E-08	$\pm 6,53E-08$	1,50E-07	$\pm 1,08E-07$
12	5,00E-08	$\pm 9,80E-08$	0	0	0	0
14	0	0	0	0	0	0

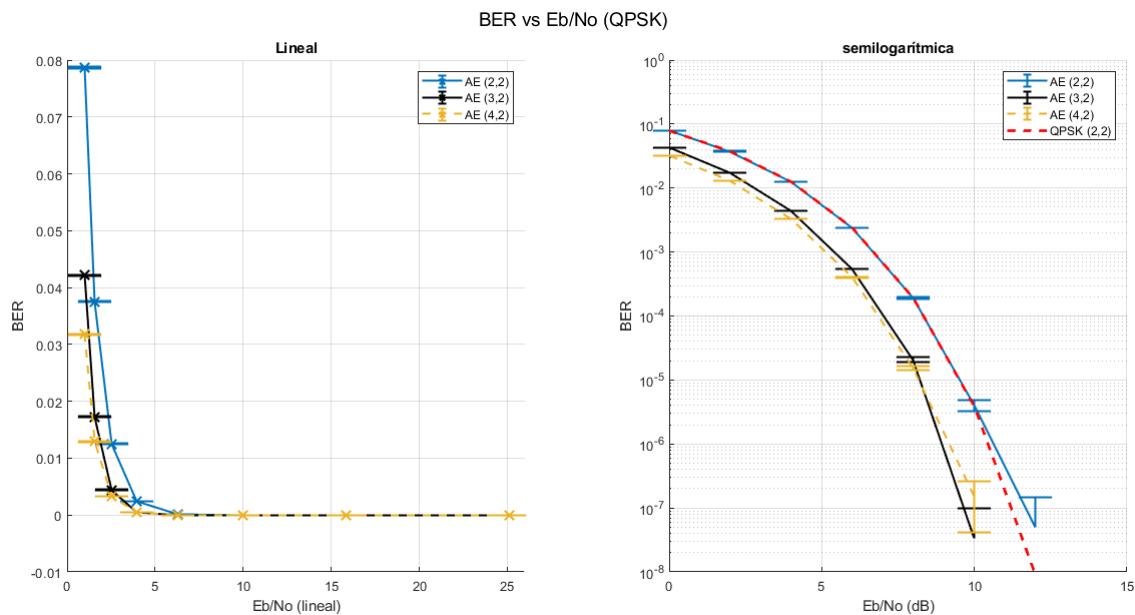


Figura 5.12: Gráfica del BLER para el auto codificador ($n,2$) normalizado en energía.

5.4. 8-PSK - auto codificador (n,3)

5.4.1. Entrenamiento

La primera configuración analizada auto codificador (2,3), con tasa de código $R = 3/2$, la cual se presenta en la Figura 5.13. Al ser la tasa mayor a 1 no es de utilidad, ya que elimina información al momento de codificar, sin embargo, se la tiene en consideración con propósitos de comparación. La tabla 5.8 presenta el tiempo de entrenamiento final para esta configuración de AE. El valor de pérdida calculado teniendo en cuenta $M = 8$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.125$:

$$L = - \sum_{i=1}^8 (y_{ij} \ln(0.125)) = 2.079 \quad (5.6)$$

El valor observado en la Figura 5.13 es ≈ 1.55 al inicio del entrenamiento y luego se estabiliza ≈ 0.8 ; por otro lado, los valores de pérdida para la validación comienza en 0.9 y se reduce en las primeras iteraciones hasta ≈ 0.3 , luego de 16 épocas. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.5 y alcanza ≈ 0.9 en el entrenamiento; sin embargo, para el conjunto de validación alcanza ≈ 1 sugiriendo que el modelo está generalizando bien los datos que son diferentes a los de entrenamiento.

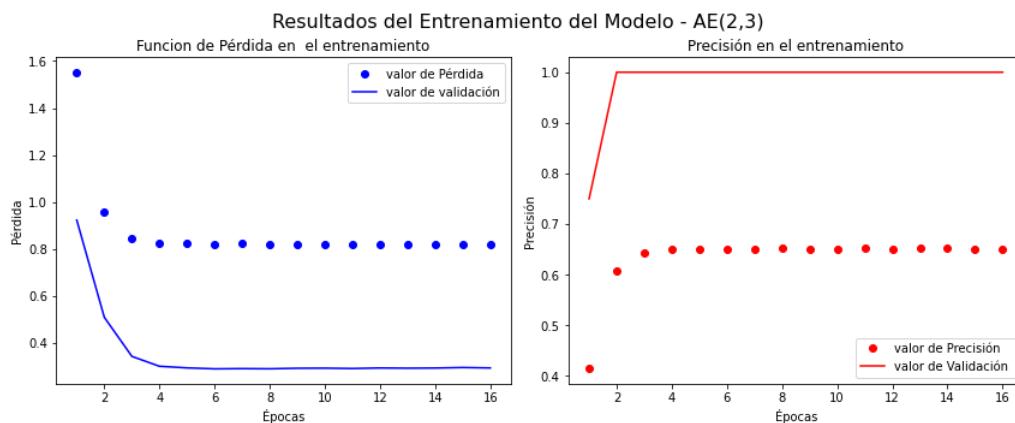


Figura 5.13: Resultados del entrenamiento para el auto codificador (2,3) normalizado en energía.

El segundo caso, presentado en la Figura 5.14, para analizar corresponde al auto codificador (3,3), que produce una tasa de código $R = 1$, es decir, sin redundancia. En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. En la Figura 5.14 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.3 y decrece hasta 0.6 en 27 épocas;

mientras, el valor de pérdida para la validación empieza en $\approx 0.5h$ y decrece hasta ≈ 0 . En la gráfica de la precisión, para el conjunto de entrenamiento, se observa que inicia con un valor de 0.6 y alcanza ≈ 0.8 . Para la validación crece hasta alcanzar ≈ 1 , sugiriendo que el modelo está generalizando bien los nuevos datos ya que tiene un número de canales n o tamaño de codificación, mayor al número de bits/símbolo (k) .

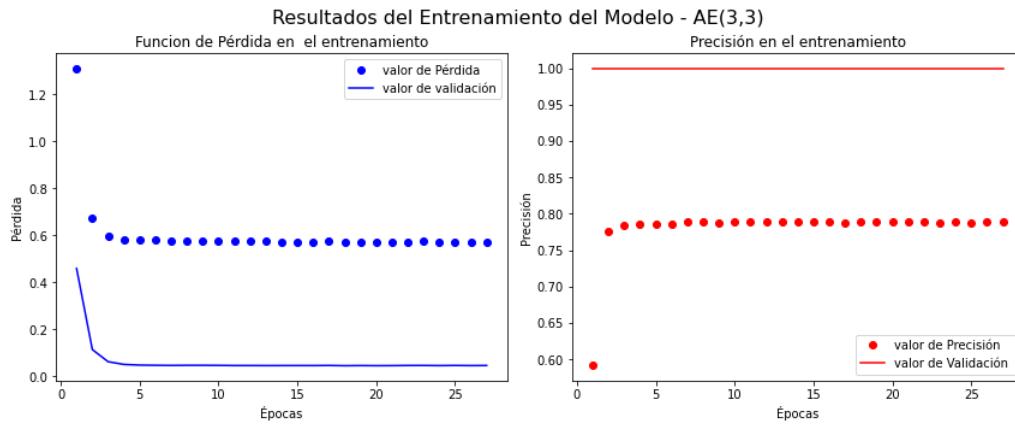


Figura 5.14: Resultados del entrenamiento para el auto codificador (3,3) normalizado en energía.

El tercer caso, presentado en la Figura 5.15, corresponde a la configuración de auto codificador (4,3), que produce una tasa de código $R = \frac{3}{4}$, es decir, la redundancia corresponde un bit más (25 %). En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. En la Figura 5.15 el valor inicial de pérdida, para el conjunto de entrenamiento, es ≈ 1.25 y decrece hasta 0.3 en 26 épocas; mientras, que el valor de pérdida para el set de validación empieza en $\approx 0.4h$ y decrece hasta ≈ 0 . En la gráfica de la precisión, para el conjunto de entrenamiento, el valor inicial es 0.55 y alcanza ≈ 0.85 (0.5 % más que para la tasa $\frac{3}{3}$) al emplear un canal adicional. Para el conjunto de validación alcanza ≈ 1 rápidamente, sugiriendo que el modelo está generalizando adecuadamente los datos, al tener un mayor número de n que k .

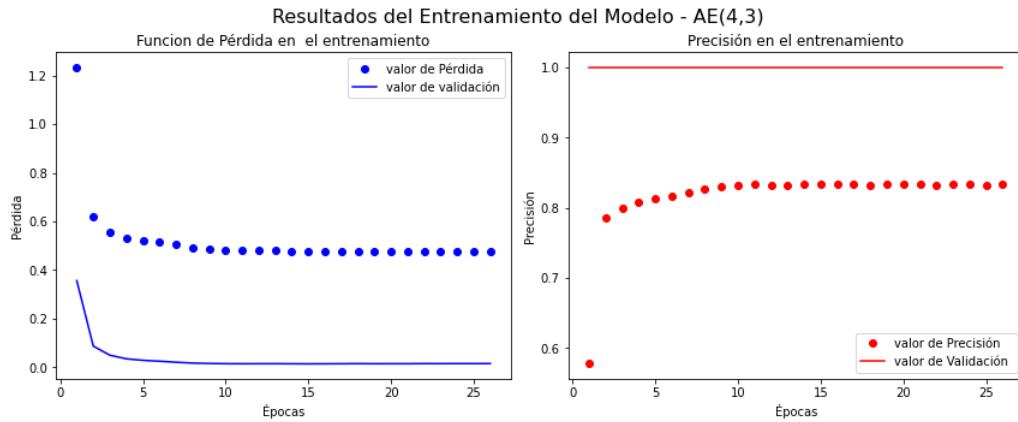


Figura 5.15: Resultados del entrenamiento para el auto codificador (4,3) normalizado en energía.

El cuarto caso, presentado en la Figura 5.16, corresponde al auto codificador (6,3), que produce una tasa de código $R = \frac{1}{2}$, es decir, se agrega redundancia del 50 %. En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. En la Figura 5.16 el valor inicial de pérdida, para el conjunto de entrenamiento, es ≈ 1.2 y decrece hasta 0.45 en 42 épocas; mientras, que el valor de pérdida para para el conjunto de validación empieza en ≈ 0.25 y decrece hasta ≈ 0 . En la gráfica de la precisión, para el conjunto de entrenamiento, el valor inicial es 0.61 y alcanza ≈ 0.84 (precisión similar que la tasa $R = \frac{3}{4}$). Para el conjunto de validación alcanza ≈ 1 rápidamente, como resultado de emplear más canales adicionales que los casos anteriores.

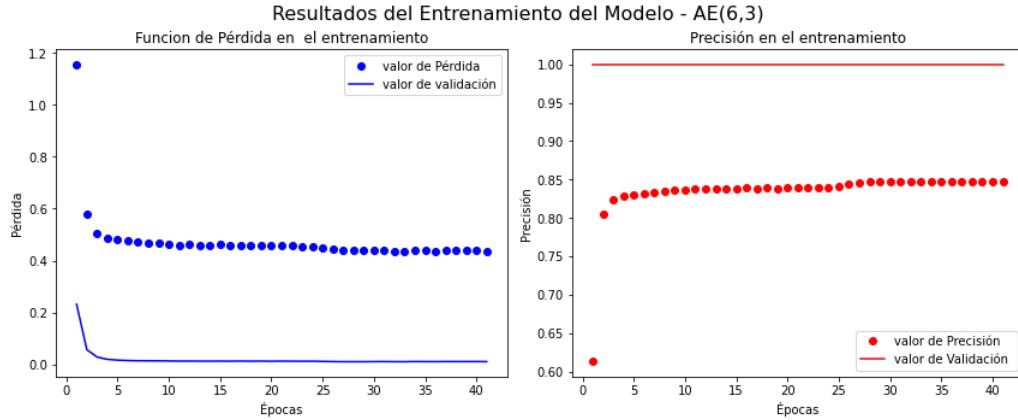


Figura 5.16: Resultados del entrenamiento para el auto codificador (6,3) normalizado en energía.

En definitiva, los mejores resultados se obtuvieron con las tasas $R = \frac{3}{4}$, y $R = \frac{1}{2}$ ya que agregan redundancia a la información. Sin embargo, este aumento en la redundancia implica

el uso de más canales y la necesidad de un mayor número de épocas para un entrenamiento efectivo, lo cual conlleva un mayor tiempo y más recursos para el procesamiento.

5.4.2. Diagrama de Constelación

En la figura 5.17 se observa que el diagrama de constelación para el auto codificador (2,3) converge a uno de 8-Phase Shift Keying (8-PSK). La gráfica de la izquierda muestra un diagrama sin la presencia de ruido; los 8 símbolos modulados se encuentran exactamente en sus posiciones ideales en el plano complejo y distribuidos uniformemente en el círculo unitario. Para un mayor detalle, en el apéndice B, tabla B.1 se observan los valores que representan a cada uno de los símbolos.

En la Figura del centro, el diagrama muestra los mismos 8 símbolos de la izquierda, pero con un nivel de SNR de 10 dB. El ruido introduce perturbaciones en la posición de cada punto, lo que causa que los puntos se dispersen alrededor de sus posiciones ideales.

Finalmente, la gráfica de la derecha muestra la constelación que corresponde a 20 dB de E_b/N_0 . El nivel de ruido es menor que en el caso anterior, resultando en una menor dispersión de los símbolos alrededor de sus posiciones ideales.

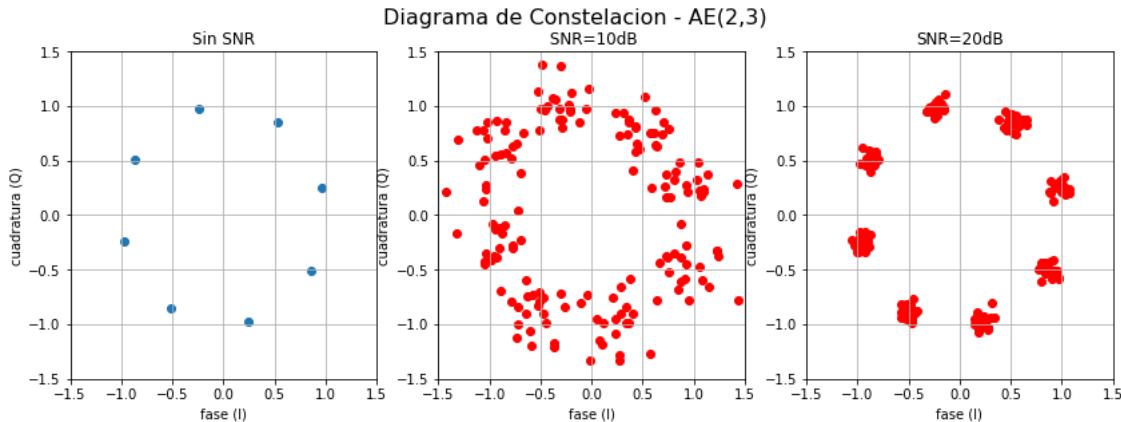


Figura 5.17: Diagrama de constelación para el auto codificador (2,3) normalizado en energía.

En la Figura 5.18 se observa la reducción de dimensionalidad de los valores del auto codificador (3,3) a 2 dimensiones, mediante los algoritmos de t-SNE y UMAP. De forma similar, en la Figura 5.19 se muestra la proyección para el auto codificador (4,3) y en la figura 5.20 para el auto codificador (6,3). Al ser una proyección, en todos los casos se puede observar una constelación diferente, pero se encuentran distribuidos alrededor del círculo unitario y tratando de mantener una distancia aproximada entre sus vecinos.

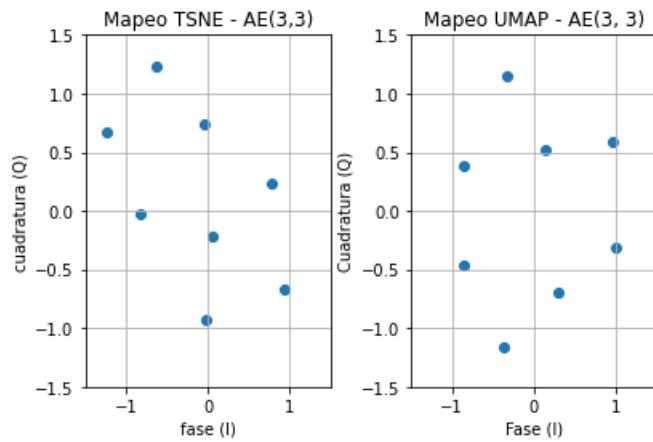


Figura 5.18: Constelaciones proyectadas para el auto codificador (3,3) normalizado en energía.

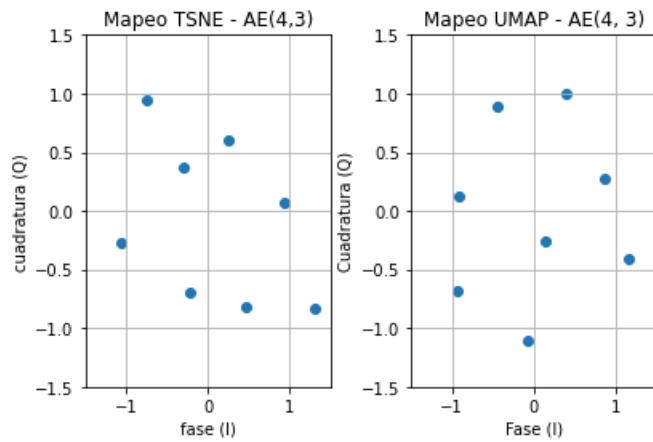


Figura 5.19: Constelaciones proyectadas para el auto codificador (4,3) normalizado en energía.

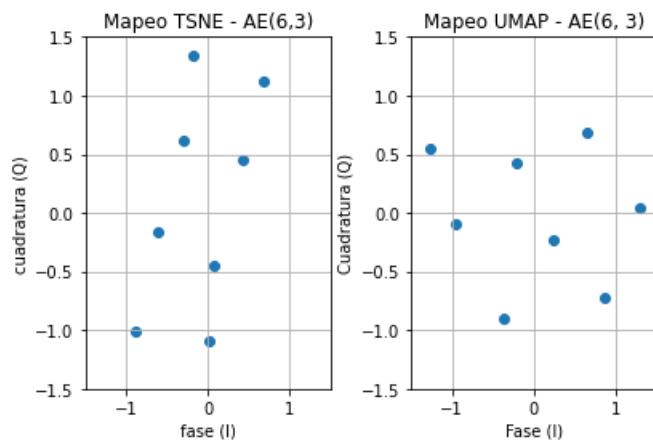


Figura 5.20: Constelaciones proyectadas para el auto codificador (6,3) normalizado en energía.

5.4.3. Probabilidad de Error

En la figura 5.21 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (n,3) y la modulación 8-PSK. La gráfica izquierda corresponde a una escala lineal, donde se aprecia cómo el BLER disminuye a medida que aumenta el E_b/N_0 para las diferentes configuraciones. Para una tasa $R = 1$ sin redundancia (AE(3,3)), se aprecia un BLER mayor que para las tasas inferiores; sin embargo, al variar la tasa de código a $R = \frac{3}{4}$ (AE(4,3)) y $R = \frac{1}{2}$ (AE(6,3)) se puede observar que tiene un rendimiento cercano. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.1. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, que muestra la relación BLER vs. E_b/N_0 en escala logarítmica, que compara el rendimiento de un sistema con auto codificador (n,3) frente a un sistema 8-PSK estándar. Inicialmente, para valores muy bajos de E_b/N_0 , el AE muestra un desempeño ligeramente inferior que el sistema tradicional (8PSK(3,3)). Sin embargo, a partir de 6 dB, la curva del AE(3,3), demuestra un mejor rendimiento. Esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información, superando al sistema 8-PSK estándar en condiciones de mayor E_b/N_0 .

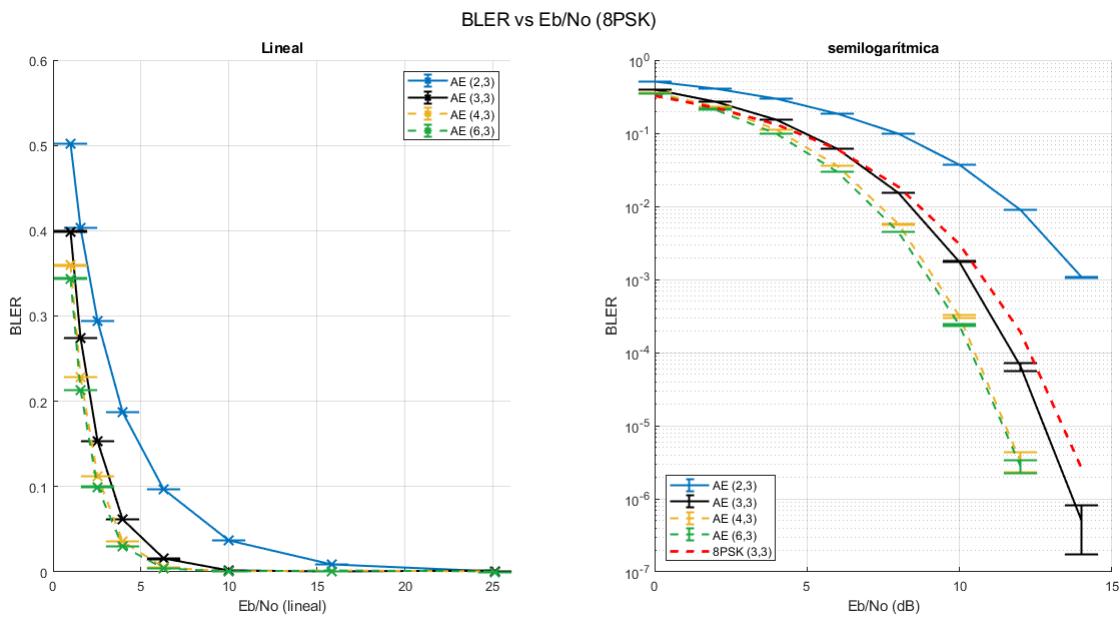


Figura 5.21: Gráfica del BLER para el auto codificador (n,3) normalizado en energía.

Al reducir la tasa de código R para agregar redundancia, el sistema es capaz de detectar y corregir errores de manera más efectiva. En la gráfica, se observa que con la tasa $R = \frac{3}{4}$

(AE(4,3), que incluye un bit de redundancia, y la tasa $R = \frac{1}{2}$ (AE(6,3)), que incluye 3 bits de redundancia, ambos alcanzan un mejor rendimiento, que un sistema 8-PSK no codificado, a partir de los 2 dB. Sin embargo, la tasa $R = \frac{1}{2}$ presenta un rendimiento superior al de la tasa $R = \frac{3}{4}$. Esto indica que el sistema logra un buen rendimiento con mayor redundancia en la codificación.

En términos de BER, en el apéndice D, tabla D.1 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales están en el rango de 10^{-4} hasta 10^{-8} , esto indica una dispersión mínima de los datos. En la Figura 5.22 se muestran las curvas en escala lineal y logarítmica, aquí se aprecia que el comportamiento del auto codificador (3,3) en el valor de E_b/N_0 más bajo tiene un rendimiento inferior que 8-PSK no codificado; sin embargo, a partir de ≈ 7 dB el sistema empieza a tener un mejor rendimiento. Para las otras tasas de codificación, se observa un mejor rendimiento que 8-PSK, en todo el rango de análisis, siendo el auto codificador (6,3) el de mejor rendimiento. Al comparar con la gráfica del BLER (figura 5.21) es visible que tiene un mejor comportamiento; las curvas de BER empiezan en un valor mas pequeño indicando que la probabilidad de error a nivel de bit es menor, para el auto codificador (4,3) es de 10^{-1} y por debajo para el (6,3). En todo el rango de E_b/N_0 el BER es menor al BLER, indicando que hay pocos bits erróneos en general, esos errores están distribuidos de manera que un solo bit erróneo puede hacer que todo un bloque sea considerado erróneo, aumentando el BLER.

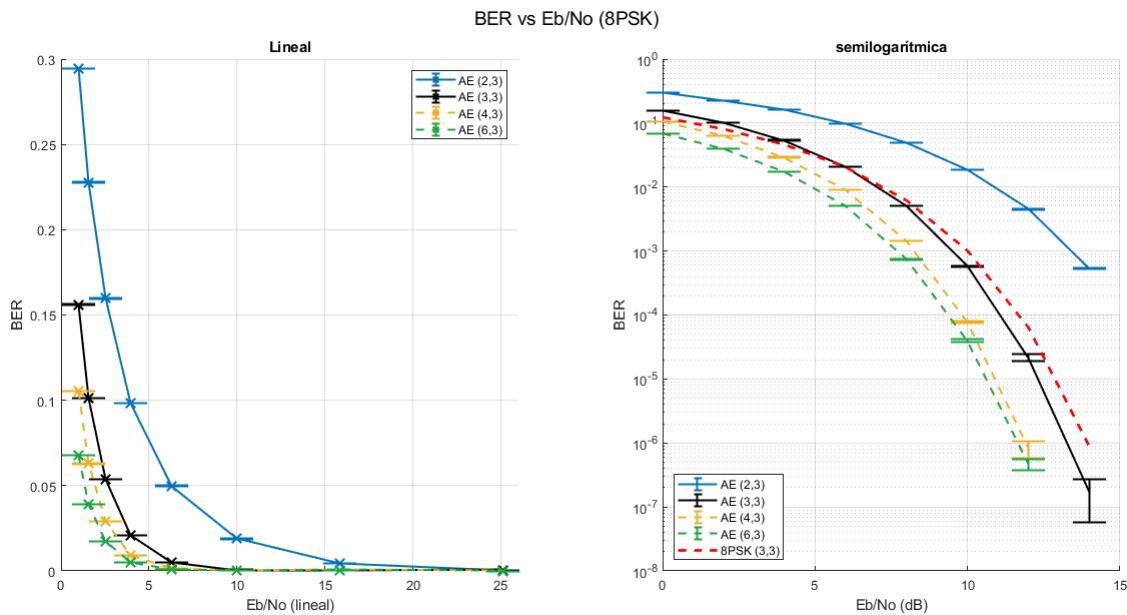


Figura 5.22: Gráfica del BER para el auto codificador (n,3) normalizado en energía.

5.5. 16-PSK - auto codificador (n,4)

5.5.1. Entrenamiento

El primer caso, figura 5.23, a considerar es el auto codificador (2,4) que genera una tasa de código $R = 2$. En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. El valor de pérdida calculado teniendo en cuenta $M = 16$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.0625$:

$$L = - \sum_{i=1}^{16} (y_{ij} \ln(0.0625)) = 2.77 \quad (5.7)$$

El valor inicial de pérdida observado en la figura 5.23 es ≈ 1.9 al inicio del entrenamiento y luego se estabiliza ≈ 1.35 ; por otro lado, los valores de pérdida para la validación comienza en 1.2 y decrece a ≈ 0.8 luego de 15 épocas. Para la gráfica de precisión, empieza con un valor de ≈ 0.3 y alcanza ≈ 0.4 en el entrenamiento; sin embargo, para el set de validación alcanza ≈ 1 después de la época 3; este comportamiento se debe al uso de tasa mayor a 1 y no es aplicada por su característica de eliminar información, pero se tendrá en cuenta para una comparación posterior y evaluar rendimientos.

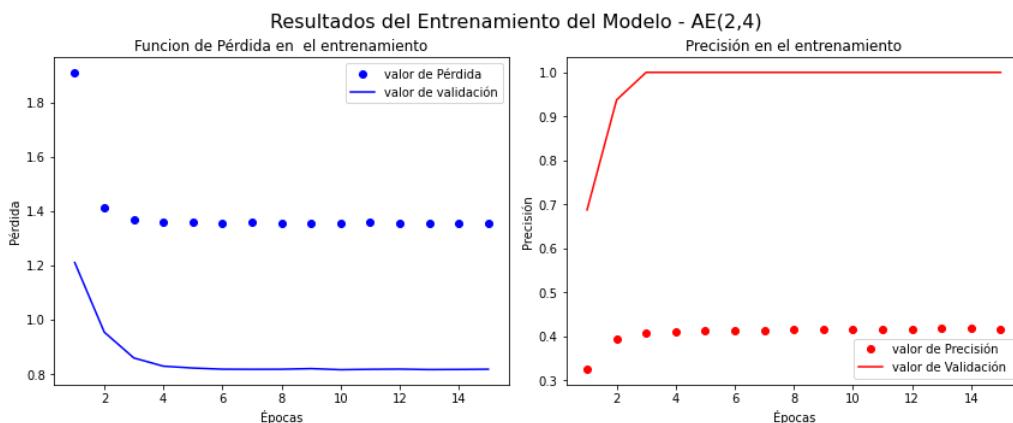


Figura 5.23: Resultados del entrenamiento para el auto codificador (2,4) normalizado en energía.

El segundo caso de análisis es el auto codificador (4,4), que produce una tasa de código $R = 1$, es decir, no se agrega redundancia. En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. En la figura 5.24 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.5 y decrece hasta 0.35 en 15 épocas; mientras, el valor de pérdida para la validación empieza en ≈ 0.4 y decrece hasta ≈ 0 . En la gráfica de la precisión, en el

entrenamiento, inicia con un valor de 0.6 y alcanza ≈ 0.87 (el doble que la tasa $R = 2$). Para la validación en un inicio alcanza ≈ 1 , este comportamiento se debe al uso de un mismo número de bits k y un mismo n .

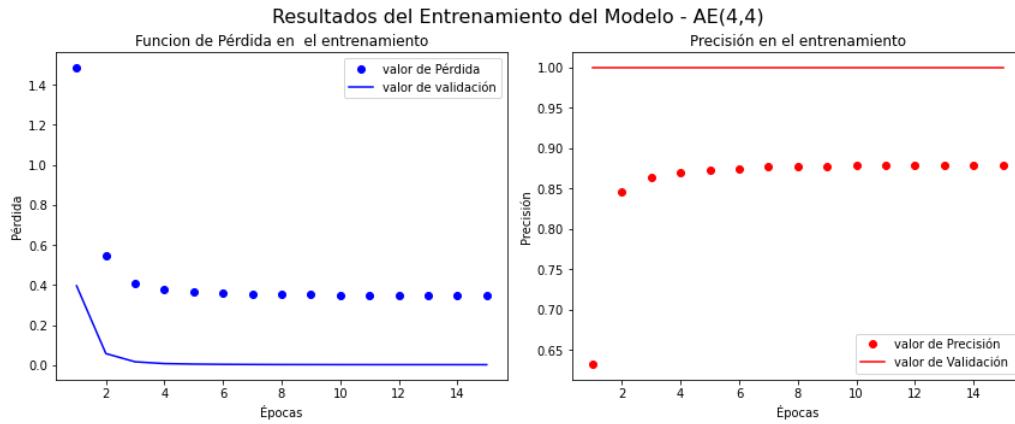


Figura 5.24: Resultados del entrenamiento para el auto codificador (4,4) normalizado en energía.

El tercer caso es para el auto codificador (6,4), que produce una tasa de código $R = \frac{2}{3}$, es decir, se agrega redundancia del 33 %. En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. En la figura 5.25 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.3 y decrece hasta 0.2 en 29 épocas; mientras, que el valor de pérdida para la validación empieza en ≈ 0.19 y decrece hasta ≈ 0 . En la gráfica de la precisión, en el entrenamiento, el valor inicial es 0.71 y alcanza ≈ 0.92 (9 % mas que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, como resultado de emplear mas canales, n , adicionales para agregar redundancia que en el caso anterior.

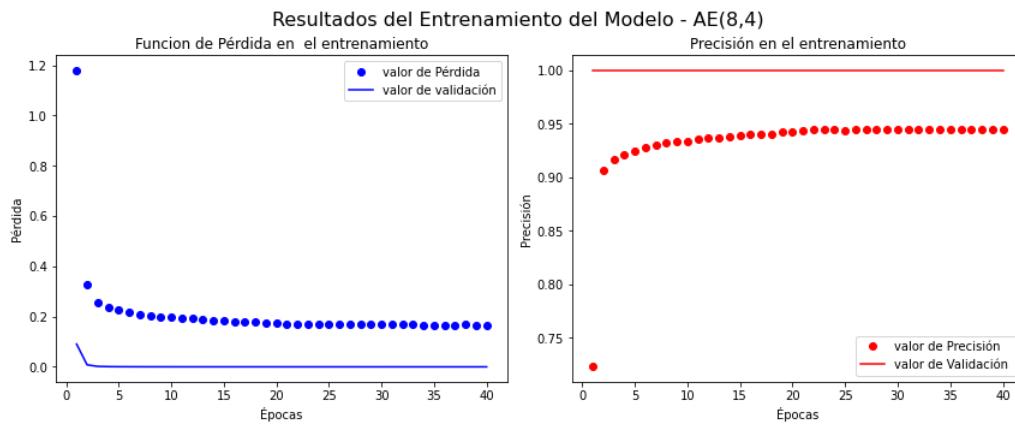


Figura 5.25: Resultados del entrenamiento para el auto codificador (6,4) normalizado en energía.

El cuarto caso es para el auto codificador $(8,4)$, que produce una tasa de código $R = \frac{1}{2}$, es decir, se agrega redundancia del 50 %. En la tabla 5.8 se puede observar el tiempo de entrenamiento final para esta tasa de código. En la figura 5.26 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.2 y decrece hasta 0.2 en 40 épocas; mientras, que el valor de pérdida para la validación empieza en ≈ 0.1 y decrece hasta ≈ 0 . En la gráfica de la precisión, en el entrenamiento, el valor inicial es 0.7 y alcanza ≈ 0.93 (2 % mas que la tasa $R = \frac{6}{4}$). Para la validación alcanza ≈ 1 rápidamente, como resultado usar mas canales, n , adicionales que los casos anteriores.

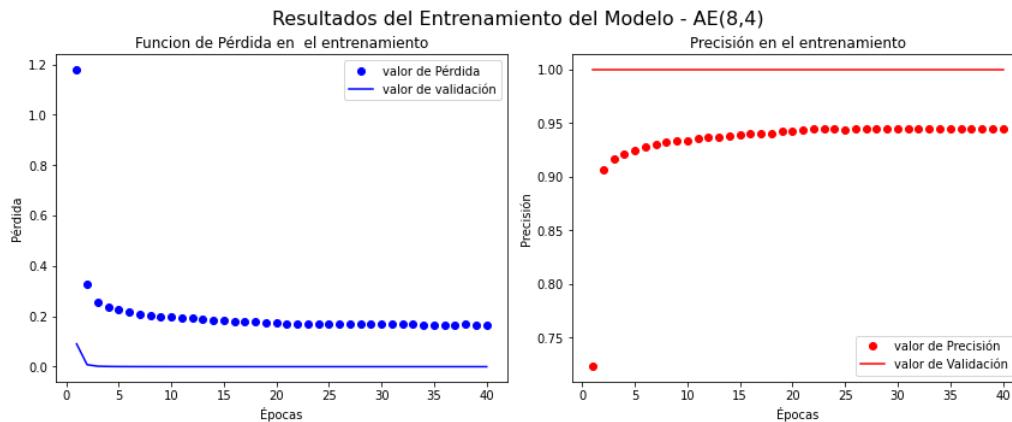


Figura 5.26: Resultados del entrenamiento para el auto codificador $(8,4)$ normalizado en energía.

En conclusión, los mejores resultados de entrenamiento se obtuvieron con las tasas de código $R = 1$, $R = \frac{2}{3}$ y $R = \frac{1}{2}$. Las dos ultimas tasas permite agregar bits de redundancia a la información, lo cual mejora la corrección de errores y en consecuencias un mejor entrenamiento. Pero, ese aumento en la cantidad de canales implica y requiere un mayor número de épocas para un entrenamiento óptimo, y en consecuencia un mayores recursos y tiempo para el entrenamiento.

5.5.2. Diagrama de Constelación

En la figura 5.27 se observan que el diagrama de constelación para el auto codificador $(2,2)$ converge en uno de 16-Phase Shift Keying (16-PSK). La figura de la izquierda muestra un diagrama sin la presencia de ruido; para este caso, las 16 señales moduladas se encuentran exactamente en sus posiciones ideales en el plano complejo y distribuidas uniformemente en el círculo unitario. Para un mayor detalle, en el apéndice B, tabla B.2 se observan los valores salida de la parte del codificador.

En la figura del centro, el diagrama muestra la misma señal modulada de la izquierda, pero

con un nivel de SNR de 10 dB. El ruido introduce perturbaciones causando que los símbolos se dispersen alrededor de sus posiciones ideales y haya interferencia con sus vecinos. Por lo tanto, la probabilidad de error tienda a incrementarse.

Por último, la figura de la derecha muestra la constelación que corresponde a un SNR de 20 dB. El nivel de ruido es menor resultando en una menor dispersión de cada uno de los 16 símbolos. Aun con la presencia de ruido, para este nivel de SNR, la demodulación de esta señales tendrá una probabilidad de error muy baja.

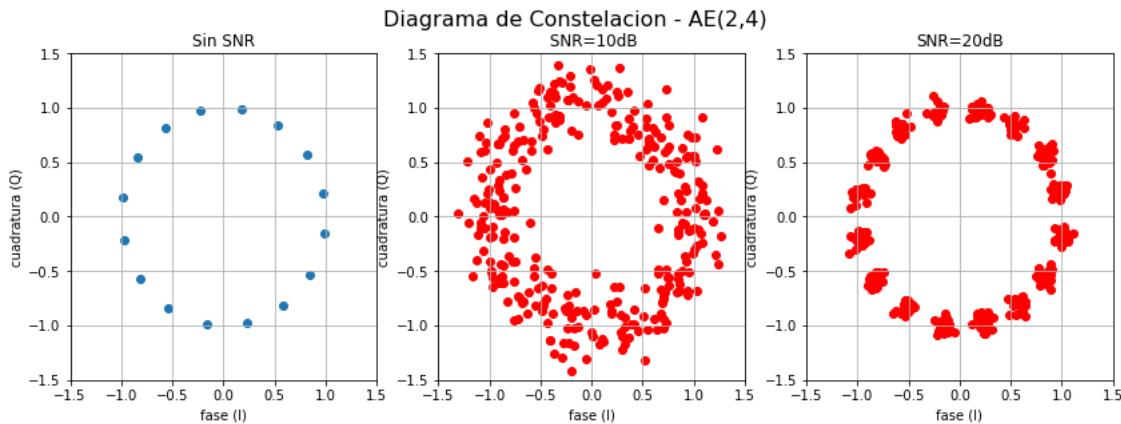


Figura 5.27: Diagrama de constelación para el auto codificador (2,4) normalizado en energía.

En la figura 5.28 se observa la proyección realiza de la constelación 4 canales para la transmisión; es decir, el tamaño del bloque a transmitir es 4 sin redundancia. Los dos diagramas muestran una distribución diferente ya que son proyecciones y depende el algoritmo empleado. Al observar los diagramas se aprecia que tratan de mantener una distancia con sus vecinos, esto ayudara a la red en el proceso de decodificación para reducir la probabilidad de error.

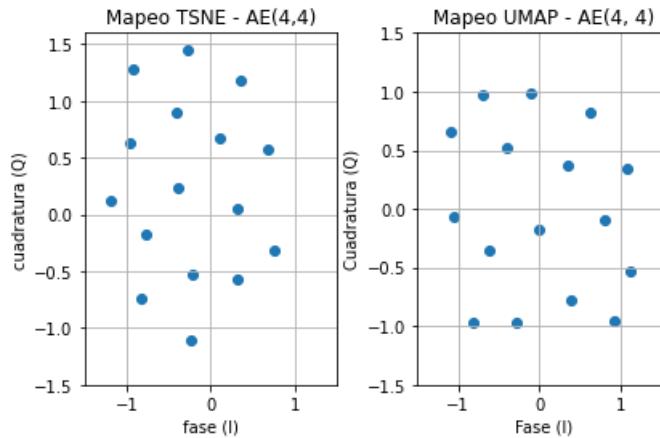


Figura 5.28: Constelaciones proyectadas para el auto codificador (4,4) normalizado en energía.

En la figura 5.29 se observa la proyección para el AE (6,4). En este caso el tamaño del bloque es de 6, con una redundancia de dos bit (33 %). En este caso se aprecia una mejor representación pero se mantiene la tendencia en los símbolos de mantenerse dentro del círculo unitario.

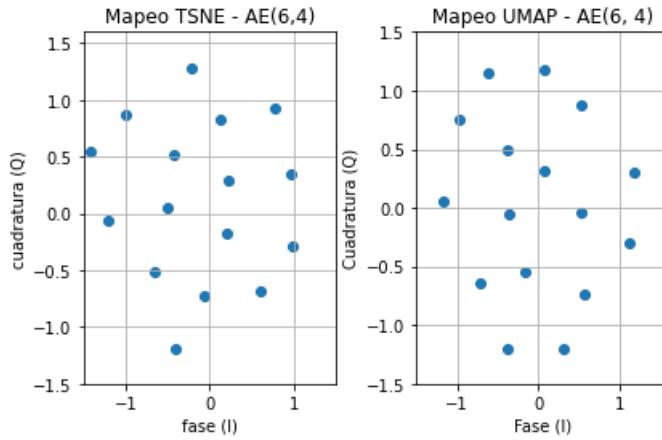


Figura 5.29: Constelaciones proyectadas para el auto codificador (6,4) normalizado en energía.

Por ultimo, en la figura 5.30 se observa la proyectadas para el AE (8,4). En este caso el tamaño del bloque es de 8, con una redundancia de 5 bits (50 %).

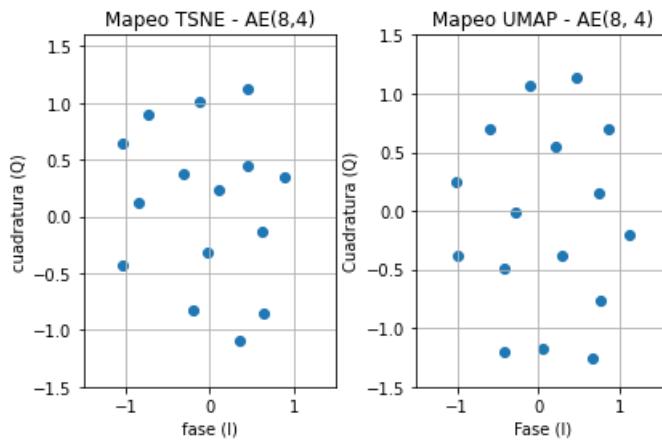


Figura 5.30: Constelaciones proyectadas para el auto codificador (8,4) normalizado en energía.

5.5.3. Probabilidad de Error

En la figura 5.31 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (n,4) y la modulación 16-PSK. La gráfica izquierda corresponde a una escala lineal, donde se aprecia cómo el BLER disminuye a medida que aumenta el E_b/N_0 para las diferentes tasas de código. Como se mencionó previamente la tasa $R = \frac{3}{2}$ no es usada por

lo que su BLER es el inferior de todos los casos. Para una tasa $R = 1$ sin codificación (AE(4,4)), se aprecia el BLER mínimo requerido para un sistema confiable. Al variar la tasa de código a $R = \frac{2}{3}$ (AE(6,4)) y $R = 1/2$ (AE(8,4)) se puede observar que tiene un rendimiento cercano, siendo mejor la segunda tasa. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.2. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, que muestra la relación BLER vs. E_b/N_0 en escala logarítmica, se compara el rendimiento de un sistema con auto codificador (n,4) frente a un sistema 16-PSK estándar. Inicialmente, para valores muy bajos de E_b/N_0 , el AE muestra un desempeño ligeramente mejor que el sistema tradicional (16PSK(4,4)). Sin embargo, la curva del AE (3,3), demuestra un mejor rendimiento. Esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información, superando al sistema 16-PSK estándar en condiciones de mayor E_b/N_0 .

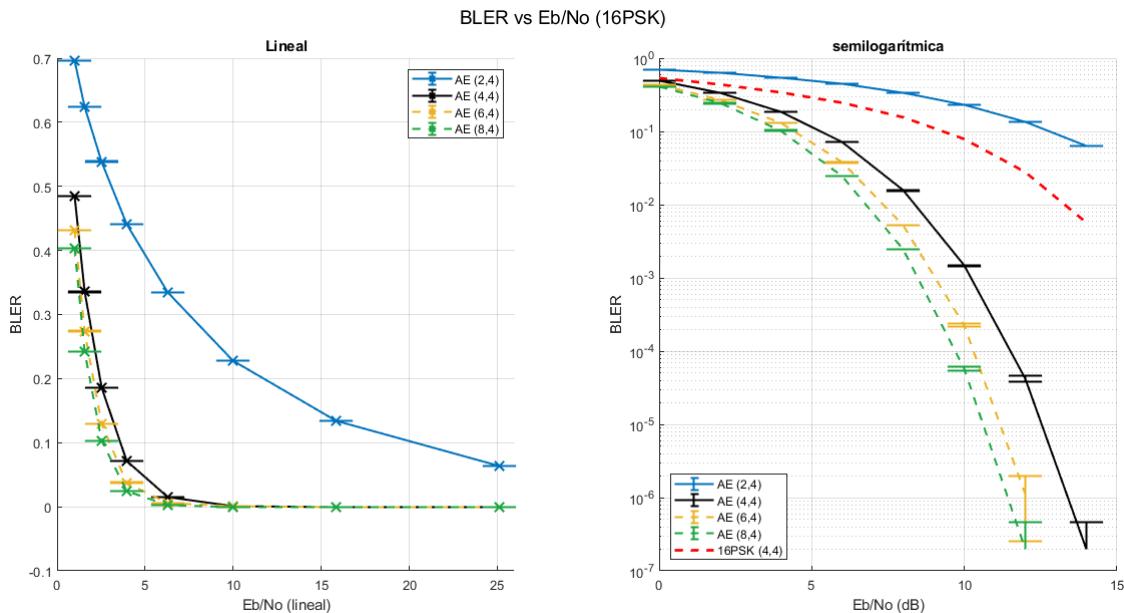


Figura 5.31: Gráfica del BLER para el auto codificador (n,4) normalizado en energía.

Al reducir la tasa de código R para agregar redundancia, el sistema es capaz de detectar y corregir errores de manera más efectiva. En la gráfica, se observa que con la tasa $R = \frac{2}{3}$ (AE(6,4)), que incluye dos bit de redundancia, y la tasa $R = \frac{1}{2}$ (AE(8,4)), que incluye 4 bits de redundancia, ambos alcanzan un mejor rendimiento, que un sistema 16-PSK sin codificación y el auto codificador (4,4). En inicio se puede observar que para niveles bajo de E_b/N_0 el rendimiento es mejor ya que el auto codificador a sido capaz de detectar y corregir errores.

Para un BLER de 10^{-6} , la tasa $R = \frac{2}{3}$ presenta una ganancia de $\approx 1dB$ y la tasa $R = \frac{1}{2}$ (AE(8,4)) una ganancia de $\approx 2dB$ en comparación con 16-PSK no codificado. Esto indica que el sistema puede lograr un buen rendimiento con 2 o 4 bits adicional para codificación, y la capacidad mejorada de decodificación.

Con tasas de codificación más pequeñas, se necesita transmitir más bits para enviar la misma cantidad de información útil. Pero esto puede reducir la eficiencia espectral, ya que se utiliza más ancho de banda para transmitir la misma cantidad de datos lo cual puede ser problemático en canales con ancho de banda limitado. Este análisis se amplia en la sección 5.12

En términos de BER, en el apéndice D, tabla D.2 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales van están en el rango de 10^{-4} hasta 10^{-8} , esto indica una dispersión mínima de los datos. En la figura 5.32 se muestran las curvas en escala lineal y logarítmica. En este caso los auto codificadores (4,4), (6,4) y (8,4) en todo el rango de Eb/No tienen un rendimiento mejor que 16-PSK no codificado; sin embargo, este último es el de mejor rendimiento. Al comparar con la gráfica del BLER (figura 5.31) es visible que tiene un mejor comportamiento; las curvas de BER empiezan en un valor más pequeño indicando que la probabilidad de error a nivel de bit es menor, 10^{-1} para el auto codificador (4,4) y por debajo para el (6,4) y (8,4). Es comportamiento también se observa en los casos anteriores indicando que la corrección de errores a nivel de bit es mejor en los niveles bajos de Eb/No y en consecuencia una menor probabilidad de error.

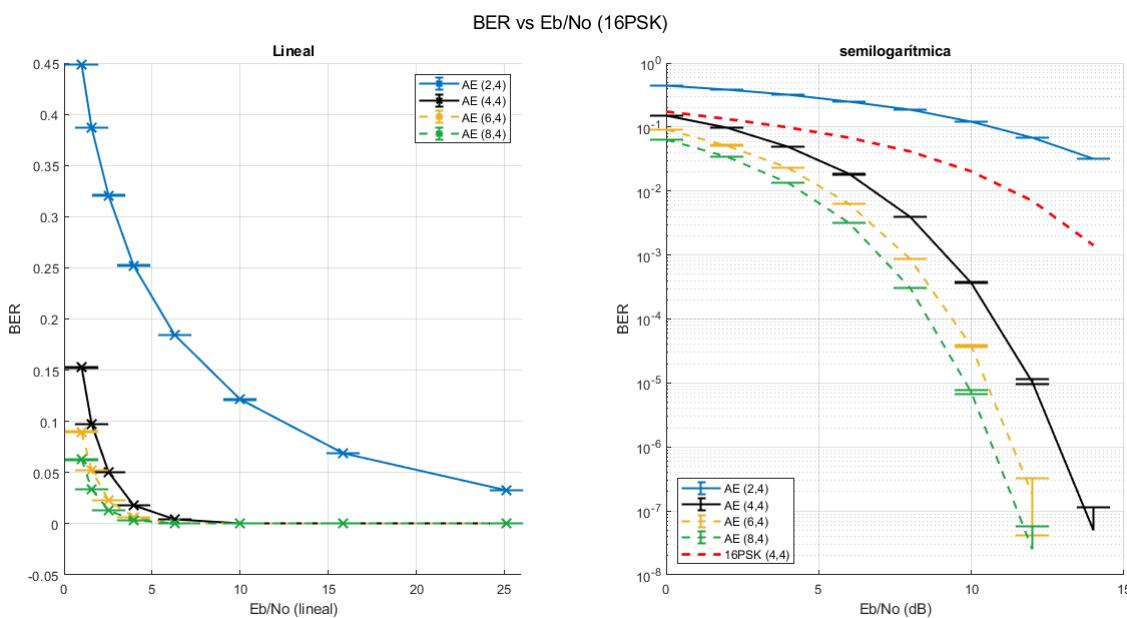


Figura 5.32: Gráfica del BER para el auto codificador (n,4) normalizado en energía.

5.6. 32-PSK - auto codificador (n,5)

5.6.1. Entrenamiento

El primer caso, figura 5.33, a considerar es el auto codificador (2,5) que genera una tasa de código $R = 2.5$. El tiempo de entrenamiento final, para esta tasa de código, se muestra en la tabla 5.8. El valor de pérdida calculado teniendo en cuenta $M = 32$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.03125$:

$$L = - \sum_{i=1}^{16} (y_{ij} \ln(0.03125)) = 3.46 \quad (5.8)$$

El valor de pérdida inicial observado en la figura 5.33 es ≈ 2.4 al inicio del entrenamiento, decrece se estabiliza ≈ 1.7 ; por otro lado, los valores de pérdida para la validación comienza en 1.7 y decrece a ≈ 1.2 luego de 22 épocas. Para la gráfica de precisión, empieza con un valor de ≈ 0.2 y alcanza ≈ 0.3 en el entrenamiento; sin embargo, para el set de validación no alcanza un valor estable y oscila entre 0.7 y 0.8; este comportamiento se debe a la tasa empleada que es mayor a 1 y se está eliminando información; es decir, la red no es capaz de ajustar los pesos para permitir minimizar los errores. Esta tasa se considera únicamente para comparación.

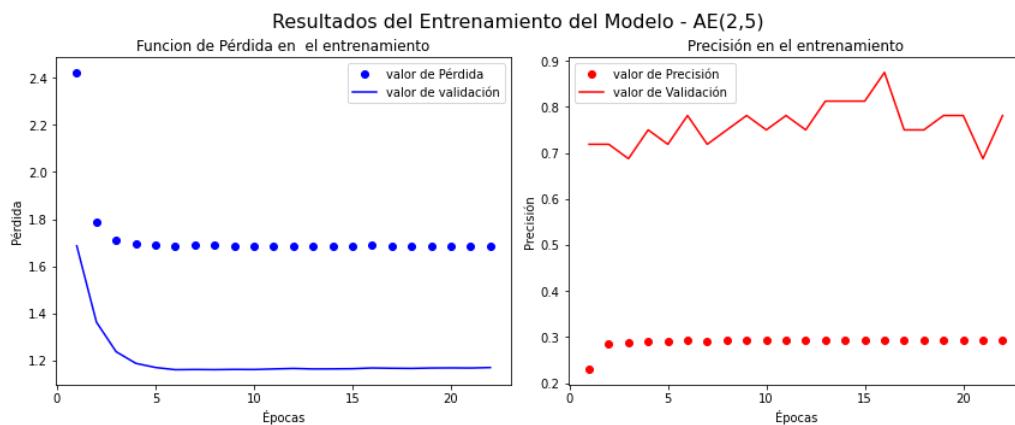


Figura 5.33: Resultados del entrenamiento para el auto codificador (2,5) normalizado en energía.

El segundo caso para análisis es para el auto codificador (5,5), que produce una tasa de código $R = 1$, es decir, no se agrega redundancia. El tiempo de entrenamiento final, para esta tasa de código, se muestra en la tabla 5.8. En la figura 5.24 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.65 y decrece hasta 0.4 en 46 épocas; mientras, el valor de pérdida para

la validación empieza en ≈ 0.2 y decrece hasta ≈ 0 . En la gráfica de la precisión, en el entrenamiento, se observa que inicia con un valor de 0.64 y alcanza ≈ 0.86 , al emplear la misma cantidad de canales y bits/símbolo. Para la validación en un inicio alcanza ≈ 1 , este comportamiento se debe a que el sistema emplea la misma cantidad de bits y canales para transmitir.

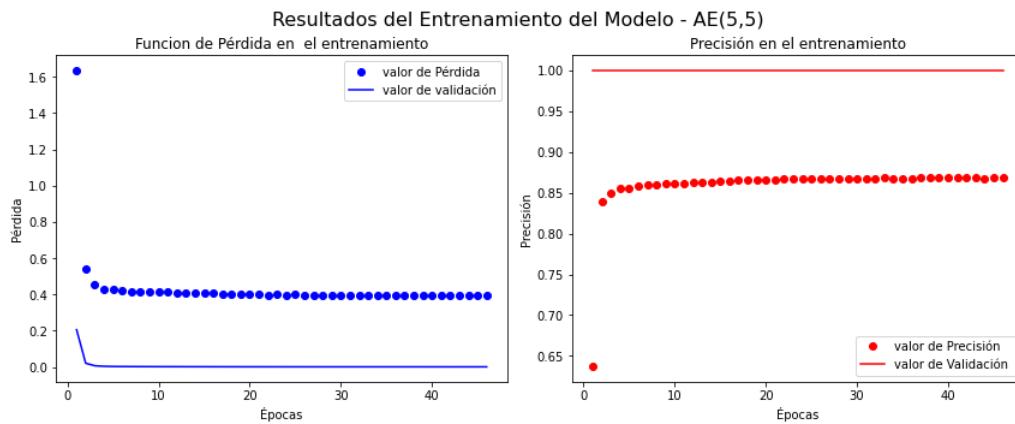


Figura 5.34: Resultados del entrenamiento para el auto codificador (5,5) normalizado en energía.

El tercer caso para análisis es el auto codificador (6,5), que produce una tasa de código $R = \frac{5}{6}$, es decir, se agrega redundancia del 16 %. El tiempo de entrenamiento final, para esta tasa de código, se muestra en la tabla 5.8. En la figura 5.35 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.5 y decrece hasta 0.3 en 22 épocas; mientras, que el valor de pérdida para la validación empieza en $\approx 0.3h$ y decrece hasta ≈ 0 . En la gráfica de la precisión, en el entrenamiento, el valor inicial es 0.7 y alcanza ≈ 0.9 (5 % mas que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, como resultado de emplear un canal adicional que el caso anterior, permitiendo un mejor ajuste en los pesos de la red y en consecuencia una mejor generalización de los datos.

El cuarto caso, es para el auto codificador (10,5), que produce una tasa de código $R = \frac{1}{2}$, es decir, se agrega redundancia del 50 %. El tiempo de entrenamiento final, para esta tasa de código, se muestra en la tabla 5.8. En la figura 5.36 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.3 y decrece hasta 0.2 en 40 épocas; mientras, que el valor de pérdida para la validación empieza en $\approx 0.1h$ y decrece hasta ≈ 0 . En la gráfica de la precisión, en el entrenamiento, el valor inicial es 0.75 y alcanza ≈ 0.94 (10 % mas que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, como resultado de emplear mas canales adicionales que los casos anteriores.

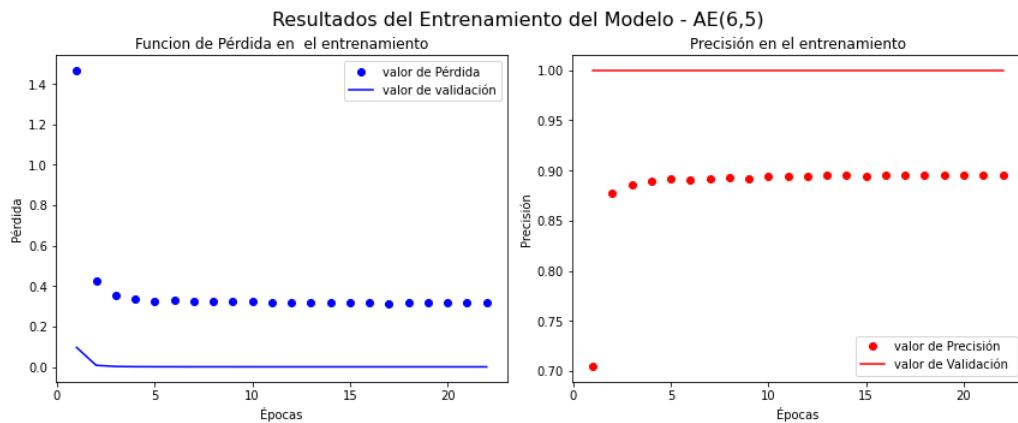


Figura 5.35: Resultados del entrenamiento para el auto codificador (6,5) normalizado en energía.

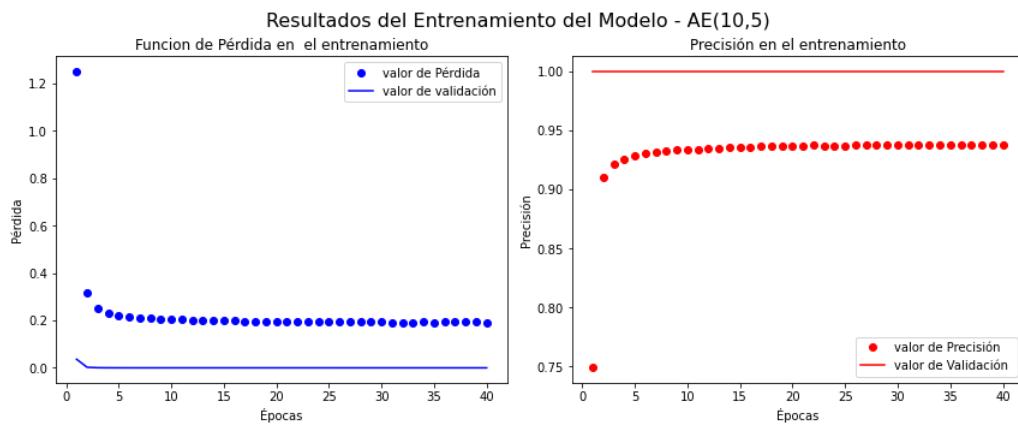


Figura 5.36: Resultados del entrenamiento para el auto codificador (10,5) normalizado en energía.

En conclusión, los mejores entrenamientos se obtuvieron con las tasas $R = 1$, $R = \frac{5}{6}$ y $R = \frac{1}{2}$ ya que permite agregar bits de redundancia a la información permitiendo a la red decodificador mejorar los datos; sin embargo, el aumento de canales genera la necesidad de una mayor número de épocas para un mejor entrenamiento, pero sera necesario una mayor tiempo y recursos para el entrenamiento.

5.6.2. Diagrama de Constelación

En la figura 5.37 se observan que el diagrama de constelación para el AE (2,5) converge en uno de 32-Phase Shift Keying (32-PSK). La figura de la izquierda muestra un diagrama sin la presencia de ruido; para este caso, las 32 señales moduladas se encuentran exactamente

en sus posiciones ideales en el plano complejo y distribuidas uniformemente en el círculo unitario. Para un mayor detalle, en el apéndice B, tabla B.3 se muestran los valores a la salida del codificador para cada símbolo.

En la figura del centro, el diagrama muestra la misma señal modulada de la izquierda, pero con un nivel de ruido añadido de 10 dB. El ruido introduce perturbaciones en la posición de cada uno de los 32 símbolos, lo que causa que estos se dispersen alrededor de sus posiciones ideales y ya empiecen a interferir con sus vecinos. Esta interferencia aumentara la probabilidad de error.

Por ultimo, la figura de la derecha muestra la constelación que corresponde a un SNR de 20 dB. El nivel de ruido es menor que en el caso de 10 dB; resultando en una menor dispersión de cada uno de los 32 símbolos alrededor de sus posiciones ideales, pero este nivel de SNR, se observa ya la posible interferencia, lo cual provoca una mayor probabilidad de error. Sin embargo, se podría emplear un mejor nivel de SNR para obtener mejor resultados.

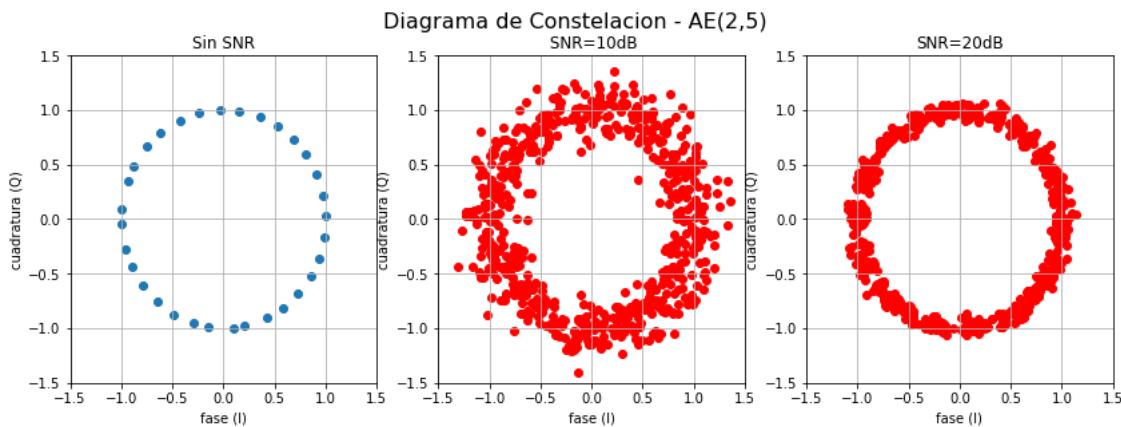


Figura 5.37: Diagrama de constelación para el auto codificador (2,5) normalizado en energía.

En la figura 5.38 se observa la proyección realiza de la constelación cuando se emplea 5 canales para la transmisión; es decir, el tamaño del bloque a transmitir es 5 y no hay redundancia. Los dos diagramas muestras una distribución diferente, pero se puede observar que tratan de mantener una distancia con sus vecinos y se encuentran al rededor del círculo unitario.

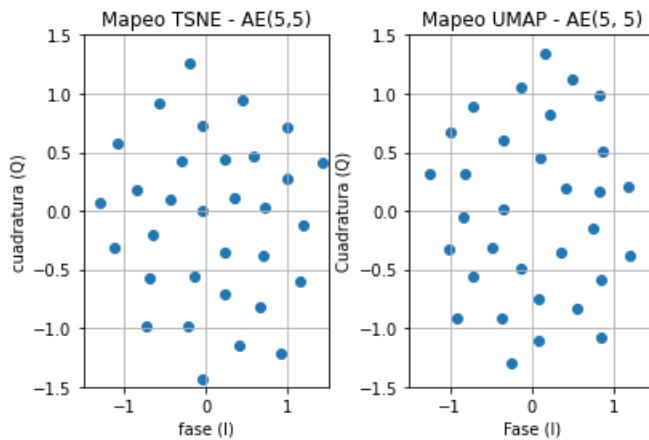


Figura 5.38: Constelaciones proyectadas para el auto codificador (5,5) normalizado en energía.

En la figura 5.39 se observa las constelaciones proyectadas para el AE (6,5). En este caso el tamaño del bloque es de 6, con una redundancia de un bit (16 %). En este caso es una representación diferente pero se mantiene la tendencia de los símbolos de mantenerse dentro del círculo unitario.

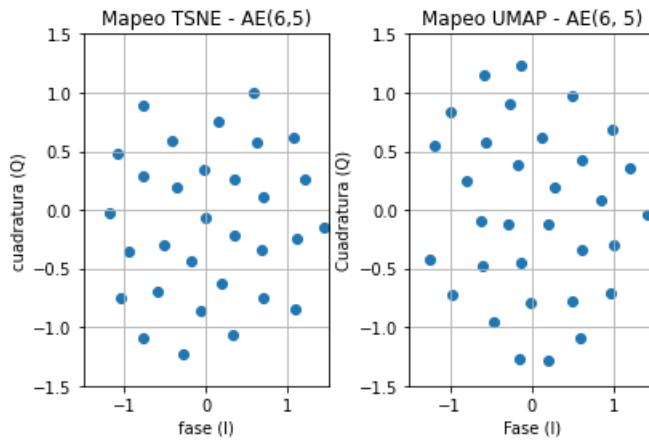


Figura 5.39: Constelaciones proyectadas para el auto codificador (6,5) normalizado en energía.

Por ultimo, en la figura 5.40 se observa las constelaciones proyectadas para el AE (10,5). En este caso el tamaño del bloque es de 10, con una redundancia de 5 bits (50 %).

5.6.3. Probabilidad de Error

En la figura 5.41 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (n,5) y la modulación 32-PSK. La gráfica izquierda corresponde a una escala lineal, donde se aprecia cómo el BLER disminuye a medida que aumenta el E_b/N_0 para

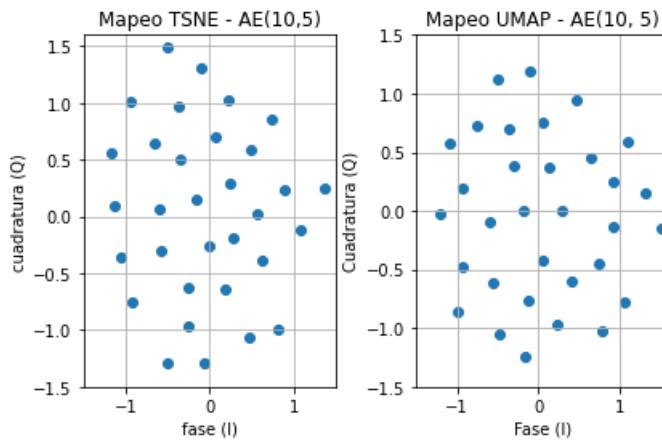


Figura 5.40: Constelaciones proyectadas para el auto codificador (10,5) normalizado en energía.

las diferentes tasas de código. Como se mencionó previamente la tasa $R = \frac{4}{2}$ no es usada por lo que su BLER es el inferior de todos los casos. Para una tasa $R = 1$ sin codificación (AE(5,5)), se aprecia el BLER mínimo requerido para un sistema confiable. Al variar la tasa de código a $R = \frac{5}{6}$ (AE(6,5)) y $R = 1/2$ (AE(10,5)) se puede observar que tiene un rendimiento cercano, siendo mejor la segunda tasa. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.3. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, que muestra la relación BLER vs. E_b/N_0 en escala logarítmica, se compara el rendimiento de un sistema con auto codificador (n,4) frente a un sistema 32-PSK estándar. Inicialmente, para valores muy bajos de E_b/N_0 , el AE muestra un desempeño ligeramente mejor que el sistema tradicional 32PSK(5,5); sugiriendo la capacidad de corrección de errores en valores de bajos de E_b/N_0 . La curva del auto codificador (3,3), demuestra un mejor rendimiento que el sistema base (32PSK(5,5)); esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información, superando al sistema 32-PSK estándar.

Al reducir la tasa de código R para agregar redundancia, el sistema puede detectar y corregir errores de manera más efectiva. En la gráfica, se observa que con la tasa $R = \frac{5}{6}$ (AE(6,5)), que incluye un bit de redundancia, y la tasa $R = \frac{1}{2}$ (AE(10,5)), que incluye 5 bits de redundancia, el rendimiento mejora para niveles bajos de E_b/N_0 debido a la capacidad del auto codificador de detectar y corregir errores. Para un BLER de 10^{-6} , la tasa $R = \frac{5}{6}$ presenta una ganancia de aproximadamente 0.5 dB y la tasa $R = \frac{1}{2}$ una ganancia de aproximadamente 1.5 dB en

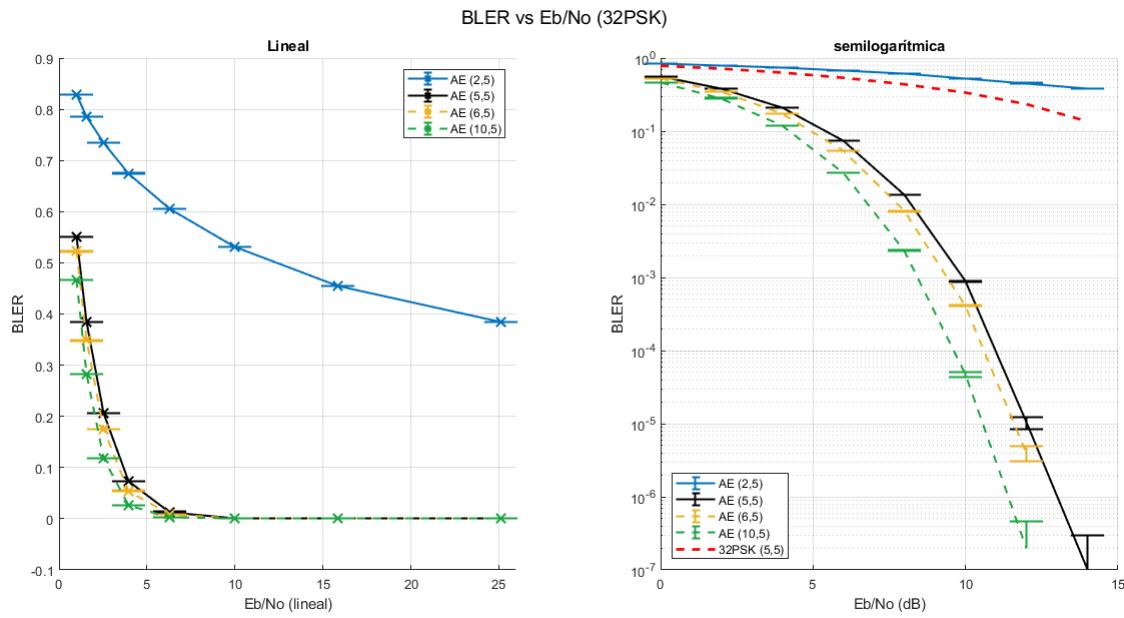


Figura 5.41: Gráfica del BLER para el auto codificador (n,5) normalizado en energía.

comparación con el auto codificador (5,5). Esto indica que el sistema puede lograr un buen rendimiento con 1 o 5 bits adicionales para codificación, gracias a la capacidad mejorada de decodificación.

Sin embargo, con tasas de codificación más bajas, se necesita transmitir más bits para enviar la misma cantidad de información útil, lo que puede reducir la eficiencia espectral. Utilizar más ancho de banda para transmitir la misma cantidad de datos puede ser problemático en canales con ancho de banda limitado. Este análisis se amplía en la sección 5.12.

En términos de BER, en el apéndice D, tabla D.3 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales van están en el rango de 10^{-4} hasta 10^{-8} , esto indica una dispersión mínima de los datos. En la figura 5.42 se muestran las curvas en escala lineal y logarítmica. En este caso los auto codificadores (5,5), (6,5) y (10,5) en todo el rango de Eb/No tienen un rendimiento mejor que 32-PSK no codificado; sin embargo, este último es el de mejor rendimiento. Al comparar con la gráfica del BLER (figura 5.41) es visible que tiene un mejor comportamiento; las curvas de BER empiezan en un valor mas pequeño indicando que la probabilidad de error a nivel de bit es menor, alrededor de 10^{-1} para el auto codificador (5,5), (6,5) y por debajo para el (10,5). Este comportamiento también se observa en los casos anteriores indicando una tendencia que los auto codificadores realizan corrección de errores, como mecanismo de mitigación de efectos del canal. Esta particularidad que es mas apreciable a nivel de bit, que la de bloque, en los niveles bajos de Eb/No . Como resultado, se genera una menor probabilidad de error.

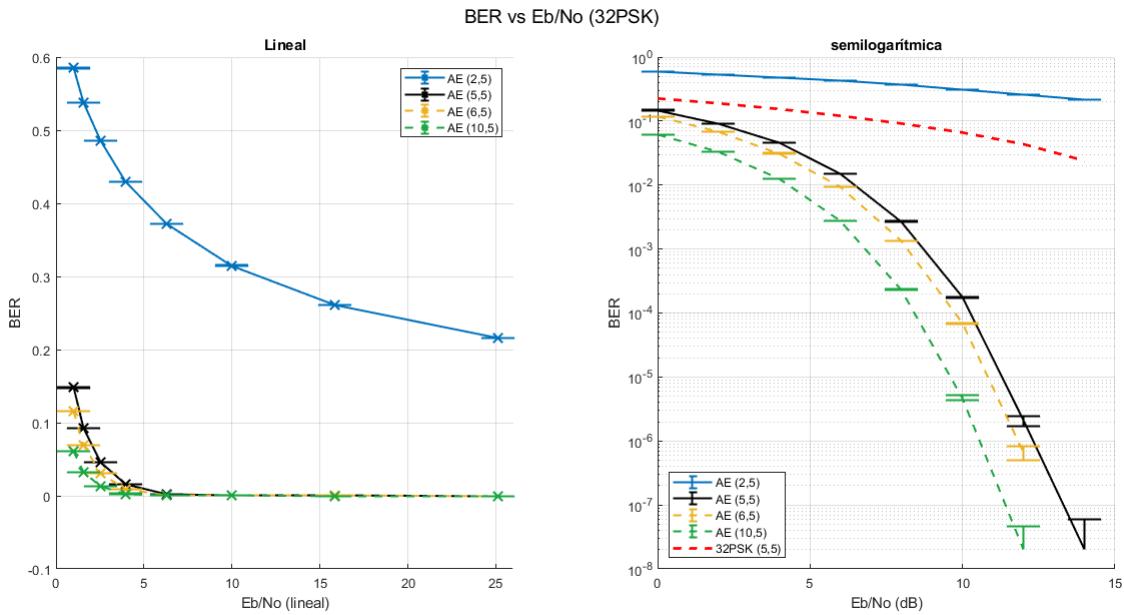


Figura 5.42: Gráfica del BER para el auto codificador (n,5) normalizado en energía.

5.7. 4-QAM - auto codificador (n,2)

5.7.1. Entrenamiento

En la tabla 5.9 se menciona el tiempo de entrenamiento para los diferentes auto codificadores con cada una de sus tasas correspondientes, así como también el número de épocas necesarias en cada uno de estos casos para generalizar el modelo evitando el sobreajuste. Finalmente, aquí también se observa el valor mínimo de pérdida el máximo en precisión para los datos de entrenamiento y validación.

El primer caso para analizar el comportamiento se trata de un auto codificador (2,2) que emplea $n = 2$ canales y $k = 2$ bits/símbolo, produciendo una tasa de código $R = 1$ (no existe redundancia). En la figura 5.43 se muestra los resultados del entrenamiento del auto codificador, empleando una descripción similar para los entrenamientos de M-PSK; el valor de pérdida calculado teniendo en cuenta $M = 4$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.25$:

$$L = - \sum_{i=1}^4 (y_{ij} \ln(0.25)) = 1.3862 \quad (5.9)$$

El valor de pérdida inicial observado es de ≈ 0.7 , que difiere del calculado, debido a las particularidades de la inicialización de pesos que en un inicio son inicializados aleatoriamente.

Luego, mientras aumenta el número de épocas la pérdida va decreciendo hasta ≈ 0.15 , este comportamiento se interpreta como una mejora en el ajuste del modelo con los datos del entrenamiento a medida que va avanzando; mientras que para el set de validación alcanza ≈ 0 luego 10 épocas, lo cual es un indicador de que el modelo ha generalizado bien sin realizar sobreajuste. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.75 y alcanza ≈ 0.95 en el entrenamiento; sin embargo, para el set de validación alcanza ≈ 1 sugiriendo que el modelo está generalizando bien los datos que no corresponden a los de entrenamiento.

Resultados del Entrenamiento del Modelo - AE(2,2)

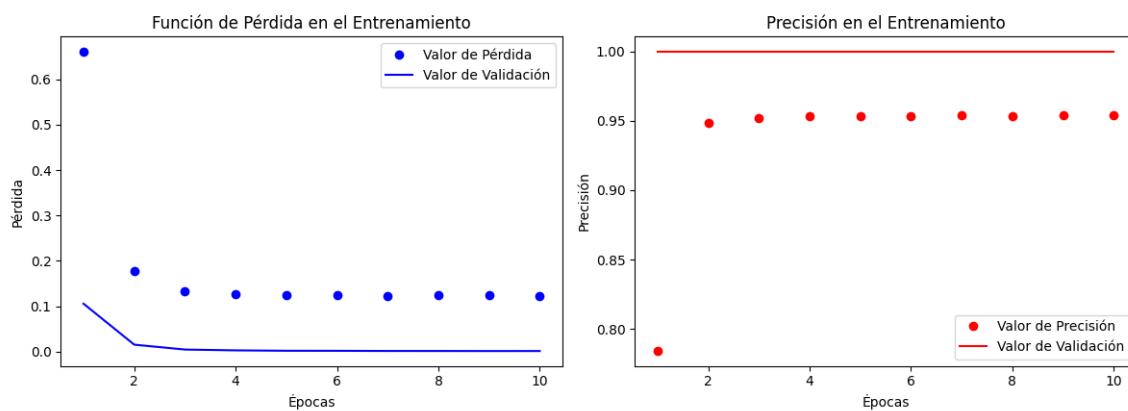


Figura 5.43: Resultados del entrenamiento para el auto codificador (2,2) normalizado en potencia.

El segundo caso (figura 5.44) es para el auto codificador (3,2) que produce una tasa $R = \frac{2}{3}$; es decir, se agrega 33 % de redundancia. El valor de pérdida inicial observado es de ≈ 0.7 y luego decrece hasta ≈ 0.09 para el entrenamiento; mientras que para el set de validación alcanza ≈ 0 con 14 épocas. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.79 y alcanza ≈ 0.98 en el entrenamiento; sin embargo, para el set de validación alcanza ≈ 1 en las 14 épocas mencionadas anteriormente.

El tercer caso (figura 5.45) para analizar es el auto codificador (4,2) que produce una tasa $R = \frac{1}{2}$; es decir, se agrega 50 % de redundancia. El valor de pérdida calculado se sabe que es de ≈ 1.3862 , pero para este caso el valor de pérdida inicial observado es de ≈ 0.6 y se estabiliza alrededor de ≈ 0.09 para el entrenamiento y ≈ 0 en la validación en 15 épocas; estos valores son similares a los analizados en la figura 5.43 para una tasa $R = \frac{2}{3}$ pero con la diferencia de que para este caso la caída se observa que es más suave desde la época 2 hasta la 4. En cuanto a la gráfica de la precisión se observa que alcanza ≈ 0.97 en el entrenamiento y ≈ 1 para la validación. La precisión en el entrenamiento tiene un pequeño incremento

Resultados del Entrenamiento del Modelo - AE(3,2)

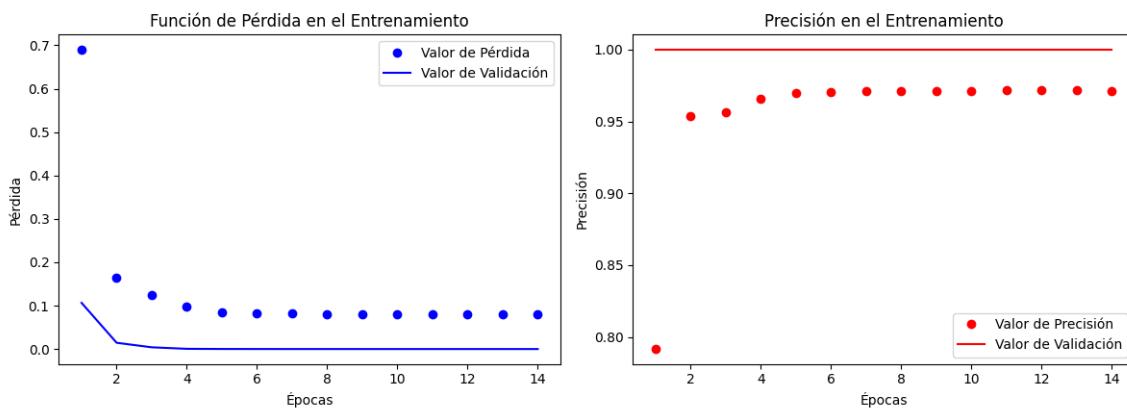


Figura 5.44: Resultados del entrenamiento para el auto codificador (3,2) normalizado en potencia.

que el caso anterior, pero además logra una estabilización en su máximo en menos épocas producto de usar 4 canales (mayor redundancia), pero la precisión en la validación es similar en los dos casos. En conclusión, el entrenamiento de estos 2 modelo ha permitido alcanzar un rendimiento similar con diferencias casi despreciables para la clasificación de las clases.

Resultados del Entrenamiento del Modelo - AE(4,2)

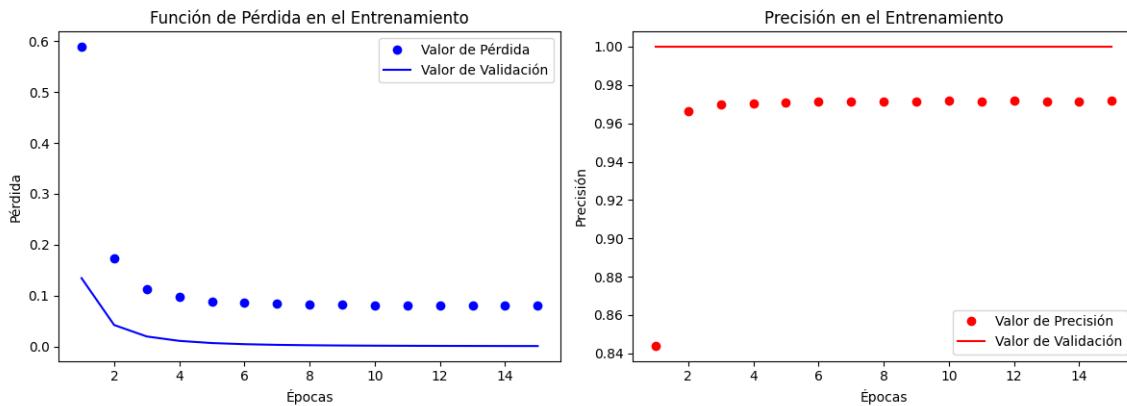


Figura 5.45: Resultados del entrenamiento para el auto codificador(4,2) normalizado en potencia.

En conclusión, se obtuvo mejores resultados con las tasas de código $R = \frac{2}{3}$ y $R = \frac{1}{2}$ que con tasa $R = 1$, ya que estas tasas permiten agregar bits de redundancia a la información, lo cual mejora la corrección de errores. Sin embargo, esto implica el aumento en la cantidad de canales por ende se requerirá un mayor número de épocas para un entrenamiento óptimo, teniendo como consecuencia el uso de mayor recursos de procesamiento lo cual se ve reflejado en

este caso con el incremento del tiempo.

5.7.2. Diagrama de Constelación

En la figura 5.46 se observa el diagrama de constelación para el auto codificador (2,2) el cual converge a una representación de la modulación 4-Quadrature Amplitude Modulation (4-QAM). La primera figura representa el diagrama de constelación sin Ruido, es decir, se presenta tal cual sale del codificador. Mientras que en las siguientes figuras se presenta el diagrama de constelación con ruido, es decir, lo que recibirá el decodificador. Para un mayor detalle, en el apéndice B, tabla B.4 se observan los valores de salida de la parte del codificador.

En la figura del centro, el diagrama de constelación presenta la adición de ruido, introduciendo el valor de SNR de 10 dB. Este efecto causa que los puntos se dispersen alrededor de sus posiciones originales, lo cual causaría confusión al momento de reconocer cada uno. En esta representación no es el caso, ya que si bien se añade un nivel de ruido, los símbolos se encuentran con una separación adecuada lo cual no permite que exista interferencia entre ellos.

Finalmente, en la última figura se muestra la constelación con un valor de SNR correspondiente a 20 dB, es decir, el valor de ruido disminuye con respecto al caso anterior. Esto resulta una menor dispersión de los puntos alrededor de las posiciones ideales de los símbolos, consiguiendo una menor interferencia de ser el caso.

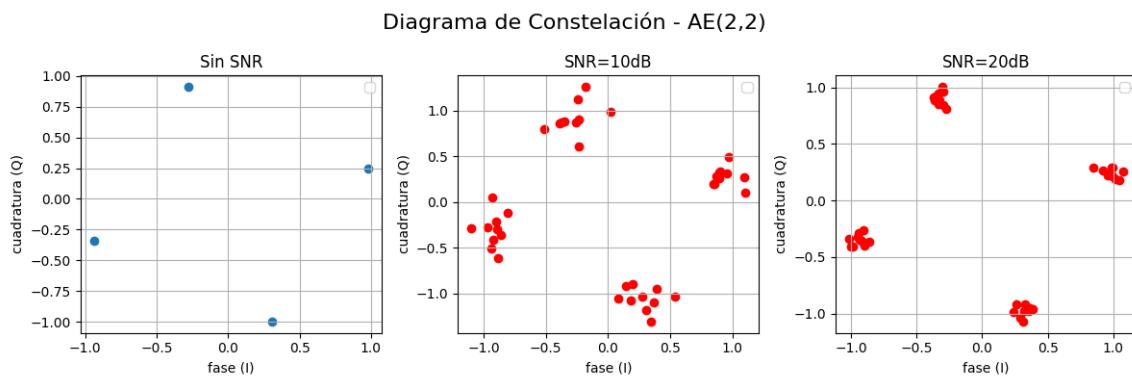


Figura 5.46: Diagrama de constelación para el auto codificador (2,2) normalizado en potencia.

Para el caso anterior, se tiene la factibilidad de que la salida del codificador este representada por 2 canales, esto hace que los símbolos estén representados con 2 valores (fase y cuadratura) para una fácil visualización. Esta facilidad no pasa cuando se usa más canales (mayor redundancia), ya que para estos casos se tiene que recurrir a reducir la dimensionalidad (proyección) a 2 dimensiones que representarán los valores de cada símbolo en el plano.

A continuación se presentan las proyecciones correspondientes a las tasas $R = \frac{2}{3}$ y $R = \frac{1}{2}$, las cuales tiene más de 2 canales a la salida del codificador.

En las figuras 5.47 se observa la reducción de dimensionalidad para 3 y 4 canales respectivamente, en las subfiguras de la izquierda se utilizó el algoritmo de t-SNE en python en el cual se observa que para 3 canales los símbolos no tienen una estructura definida como es el caso de 2 canales, para el caso de 4 canales la estructura se asemeja a una cuarta parte de una circunferencia lo cual igualmente difiere en un gran porcentaje al caso de 2 canales. Mientras que para las subfiguras de la derecha se utilizó el algoritmo de UMAP en python, el cual ofrece un resultado muy similar al caso de la constelación observada en la figura 5.46 para el caso de 3 canales, mientras que para el caso de 4 canales la estructura presenta una ligera rotación de los puntos. Estas diferencias que se observan es debido a como cada algoritmo toma los datos para representarlos en la dimensión 2, cabe recalcar que los valores mostrados en las diferentes proyecciones no representan los valores originales en los cuales se encuentras los símbolos para dimensiones mayores a 2.

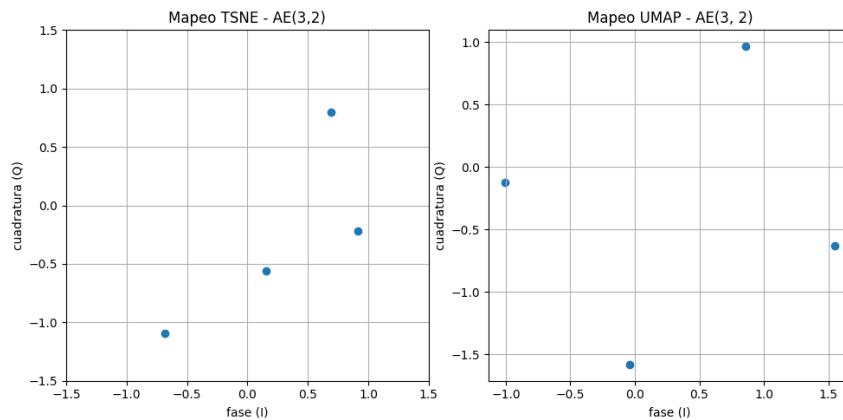


Figura 5.47: Constelaciones proyectadas para el auto codificador (3,2) normalizado en potencia.

5.7.3. Probabilidad de Error

En la figura 5.49 se presenta la gráfica del BLER con respecto a la Relación Señal-Ruido por bit (E_b/N_0). En la parte de la izquierda se presenta la gráfica a una escala lineal, AE(2,2) (tasa $R = 1$), es la que mayor valor presenta de las 3 curvas; esto es razonable ya que está no presenta ninguna redundancia, además se observa que los intervalos de confianza se hacen más pequeños conforme aumenta el E_b/N_0 , es decir, para valores más altos la dispersión de datos es casi nula, esto sucede para las 3 representaciones del auto codificador. Al variar la

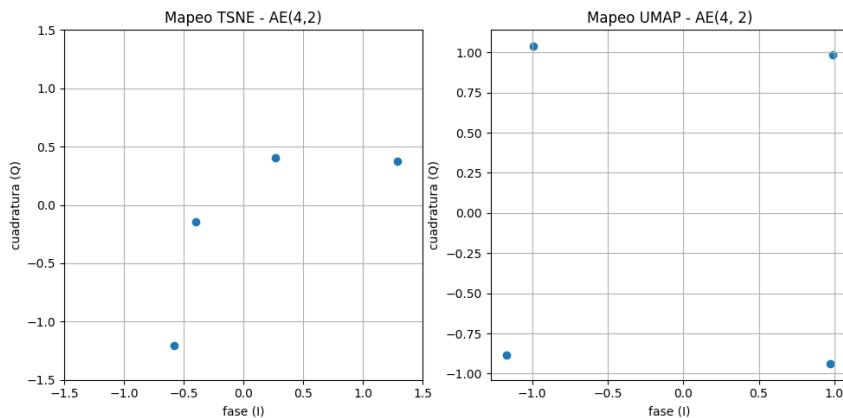


Figura 5.48: Constelaciones proyectadas para el auto codificador (4,2) normalizado en potencia.

tasa de código $R = \frac{2}{3}$ (AE(3,2)) y la tasa $R = \frac{1}{2}$ (AE(4,2)) se observa un comportamiento similar en ambos casos. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.4. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

La gráfica de la derecha corresponde a la representación del BLER en escala logarítmica, esta vez se realiza una comparación del rendimiento de auto codificador (n,2) respecto a un sistema 4-QAM no codificado. Es así que en la figura 5.49 en la parte izquierda se observa un comportamiento similar entre el AE con tasa $R = 1$ (AE(2,2)) y la curva teórica (4-QAM(2,2)), con estos resultados se intuye que esta codificación al no poseer redundancia recupera los datos transmitidos con una misma probabilidad de error que en una modulación 4-QAM sin codificación. Para el caso de las gráficas con bits de redundancia tasa $R = \frac{2}{3}$ que posee un bit de redundancia (AE(3,2)) y tasa $R = \frac{1}{2}$ que posee 2 bits de redundancia (AE(4,2)) poseen un comportamiento parecido en cuanto a la visualización de la caída de la curva, en cuanto al intervalo de confianza se presentan diferencias y esto se debe a que al momento de calcularlos los datos con los que se cuentan son la mayor parte ceros y eso hace que los intervalos tengan estos comportamientos.

A continuación se analizará el comportamiento del BER para este auto codificador (n,2). En el apéndice D, tabla D.4 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales van están en el rango de 10^{-4} hasta 10^{-8} , esto indica una dispersión mínima de los datos. En la figura 5.50 se muestran las curvas en escala lineal y logarítmica. Tanto el auto codificador (2,2) como el sistema 4-QAM sin codificación (igual que en el caso de BLER), tienen un rendimiento similar. Para las otras tasas de codificación se observa un

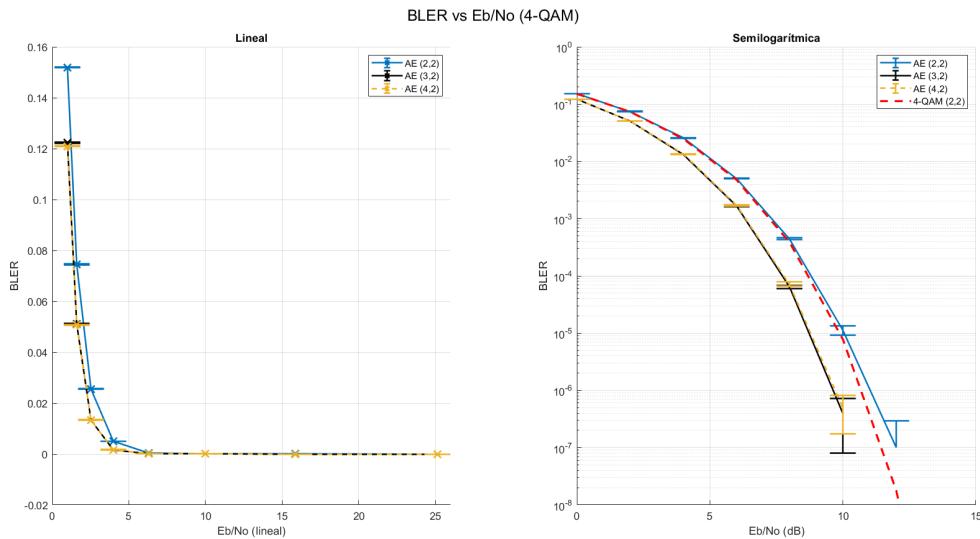


Figura 5.49: Gráfica del BLER para el auto codificador (n,2) normalizado en potencia.

mejor rendimiento que 4-QAM convencional, siendo el auto codificador (4,2) el que posee un mejor rendimiento hasta un $E_b/N_o \approx 8\text{dB}$ (diferencia mínima con (3,2)). Al comparar estos resultados con la gráfica de BLER (figura 5.49) es visible que las curvas de BER empiezan en un valor más bajo indicando que la probabilidad de error a nivel de bit es menor, por debajo de 10^{-1} . Si el BER es menor al BLER se puede llegar a la conclusión que hay pocos bits erróneos en general, esos errores están distribuidos de manera que afectan a muchos bloques y en consecuencia el BLER aumenta.

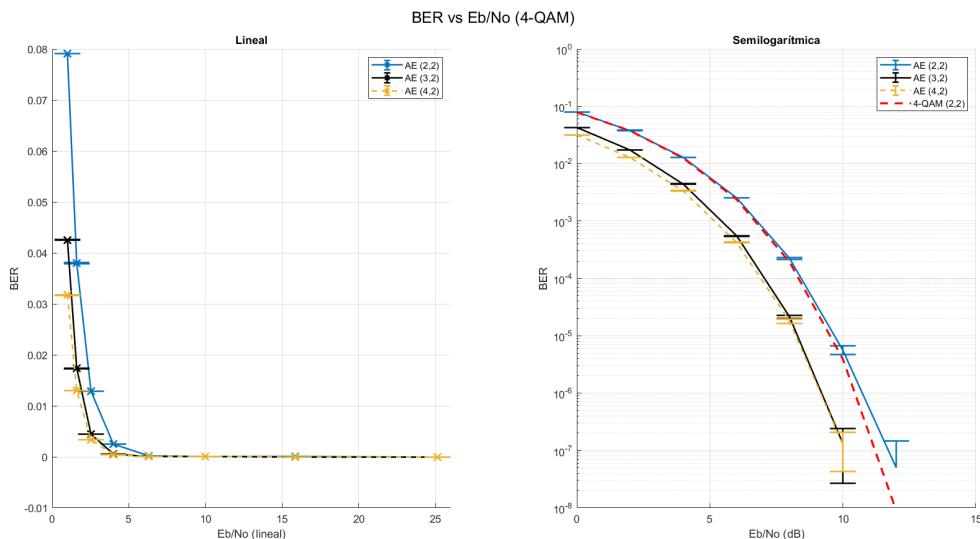


Figura 5.50: Gráfica del BER para auto codificador (n,2) normalizado en potencia.

5.8. 16-QAM - auto codificador (n,4)

5.8.1. Entrenamiento

El primer caso para analizar el comportamiento se trata de un auto codificador (2,4) que emplea $n = 2$ canales y $k = 4$ bits/símbolo, produciendo una tasa de código $R = 2$ siendo mayor a 1 esta tasa no es valida usar ya que se estaría eliminando información al momento de codificar, pero se tendrá en cuenta ya que para términos del diagrama de constelación plasma una mejor visualización. En la figura 5.51 se muestra los resultados del entrenamiento del AE, el valor de pérdida calculado teniendo en cuenta $M = 16$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.0625$:

$$L = - \sum_{i=1}^{16} (y_{ij} \ln(0.0625)) = 2.77 \quad (5.10)$$

El valor de pérdida inicial observado es de ≈ 1.7 , que difiere del calculado, debido a las particularidades de la inicialización de pesos que en un inicio son inicializados aleatoriamente. Luego, mientras aumenta el número de épocas la pérdida va decreciendo y se estabiliza en ≈ 1.2 , mientras que para el set de validación alcanza ≈ 0.4 luego de 13 épocas, este comportamiento se debe al uso de tasa mayor a 1 es por esa razón que no es aplicada en los auto codificadores ya que en vez de crear redundancia elimina información. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.4 y alcanza ≈ 0.55 en el entrenamiento; sin embargo, para el set de validación alcanza ≈ 1 .

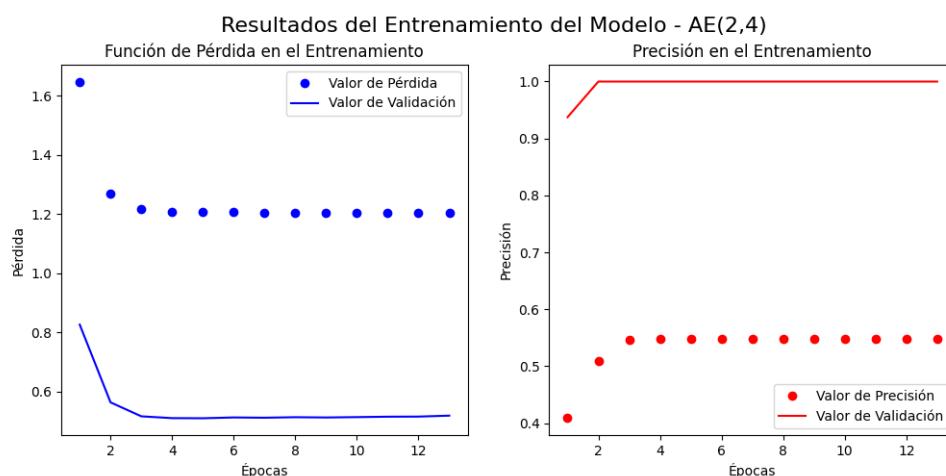


Figura 5.51: Resultados del entrenamiento para el auto codificador (2,4) normalizado en potencia.

El segundo caso figura 5.52 es para el auto codificadores (4,4) que produce una tasa $R = 1$ (sin redundancia). El valor observado de pérdida inicial es de ≈ 1.2 y luego decrece hasta ≈ 0.75 para el entrenamiento; mientras que para el set de validación alcanza ≈ 0 , menor pérdida que con $R = 2$ pero 5 épocas más que para el caso anterior. En cuanto a la gráfica de precisión, empieza con un valor de ≈ 0.6 y alcanza ≈ 0.75 en el entrenamiento (mejor precisión que con $R = 2$, sin redundancia pero tampoco sin bits de pérdida); sin embargo, para el set de validación alcanza ≈ 1 en las 19 épocas mencionadas anteriormente.

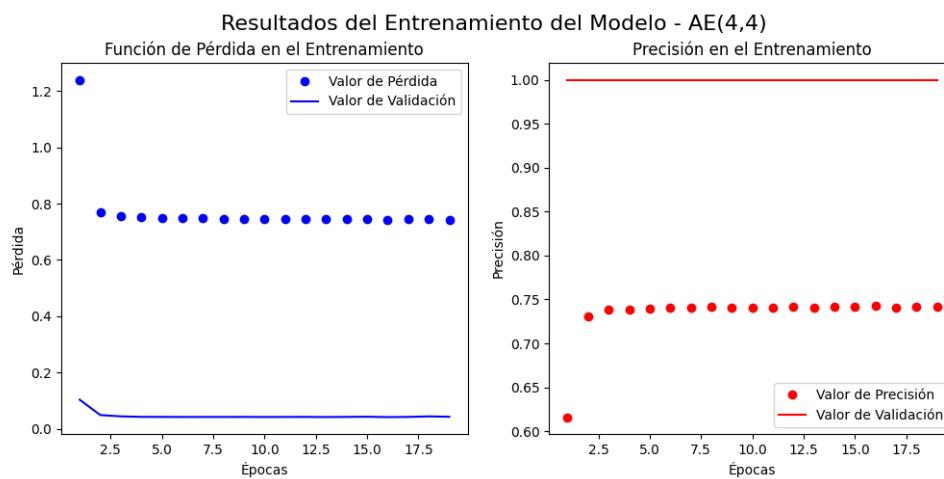


Figura 5.52: Resultados del entrenamiento para el auto codificador (4,4) normalizado en potencia.

El tercer caso el auto codificadores (6,4), que produce una tasa $R = \frac{2}{3}$; es decir, se agrega 33 % de redundancia. En la figura 5.53) el valor de pérdida inicial observado es de ≈ 1.1 y se estabiliza alrededor de ≈ 0.59 para el entrenamiento y ≈ 0 en la validación en 16 épocas, existe menor pérdida que en los casos anteriores, para $R = 2$, la pérdida a disminuido en un 50 %, en cuanto al conjunto de entrenamiento. Mientras que en la gráfica de la precisión se observa que alcanza ≈ 0.82 en el entrenamiento y ≈ 1 para la validación. La precisión en el entrenamiento tiene un pequeño incremento que en el segundo caso, pero además logra una estabilización en su máximo en menos épocas producto de usar 4 canales (mayor redundancia), pero la precisión en la validación es similar en los dos casos.

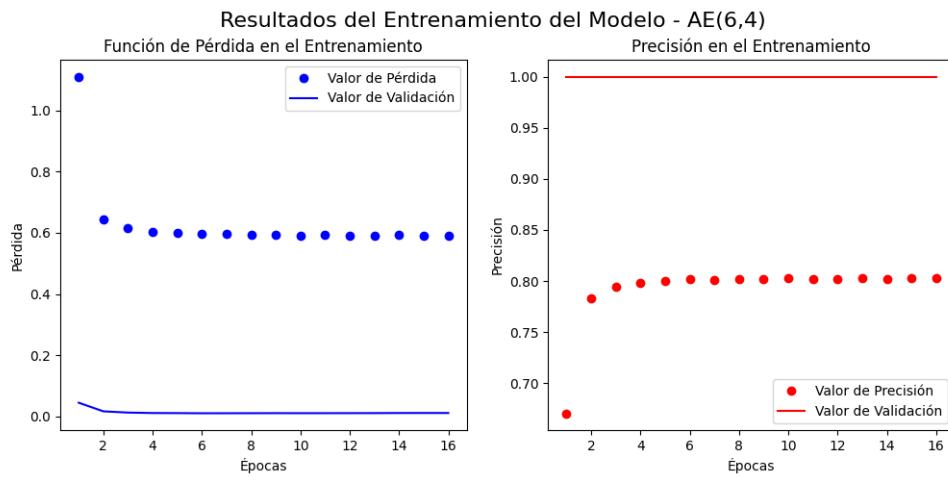


Figura 5.53: Resultados del entrenamiento para el auto codificador (6,4) normalizado en potencia.

El cuarto caso es para el auto codificadores (8,4), que produce una tasa de código $R = \frac{1}{2}$, es decir, se agrega redundancia del 50 %. En la figura 5.54 el valor inicial de pérdida, en el entrenamiento, es ≈ 1 y decrece hasta 0.5 en 21 épocas; mientras, que el valor de pérdida para la validación empieza en ≈ 0.05 y decrece hasta ≈ 0 (se reduce a la mitad en cuanto a la figura 5.26 las cuales tienen el mismo número de símbolos y canales, en este caso obtiene mejores resultados ocupando un menor número de épocas, esto sucede debido a la normalización implementada (potencia). En la gráfica de la precisión, en el entrenamiento, el valor inicial es 0.65 y alcanza ≈ 0.84 (4 % más que la tasa $R = \frac{6}{4}$). En la validación alcanza ≈ 1 rápidamente, como resultado de usar más canales.

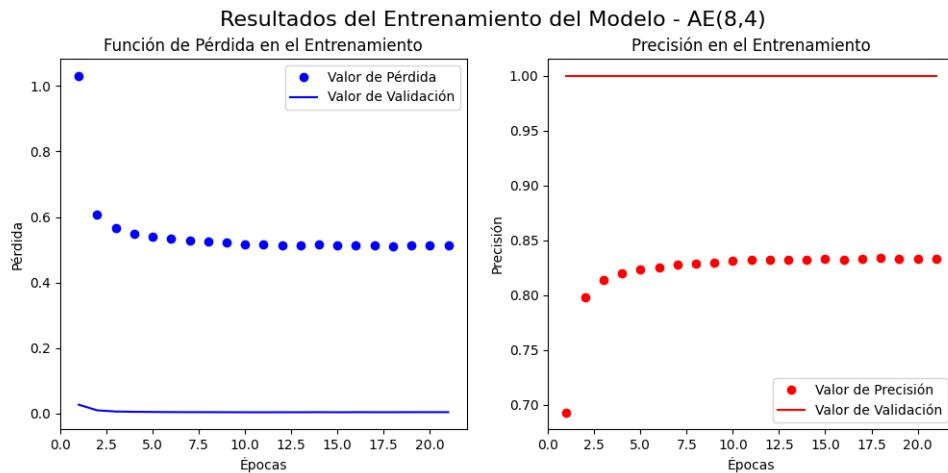


Figura 5.54: Resultados del entrenamiento para el auto codificador (8,4) normalizado en potencia.

En conclusión, se obtuvo mejores resultados con las tasas de código $R = \frac{2}{3}$ y $R = \frac{1}{2}$ que con tasa $R = 1$ y más aún con $R = 2$, ya que estas tasas permiten agregar bits de redundancia a la información, lo cual mejora la corrección de errores. Sin embargo, esto implica el aumento en la cantidad de canales por ende se requerirá un mayor número de épocas para un entrenamiento óptimo, teniendo como consecuencia el uso de mayor recursos de procesamiento lo cual se ve reflejado en este caso con el incremento del tiempo como se muestra en la tabla 5.9.

5.8.2. Diagrama de Constelación

En la figura 5.55 se observa que el diagrama de constelación para el auto codificador (n,4) converge en uno de gls16-QAM. La figura de la izquierda muestra un diagrama sin la presencia de ruido (salida del codificador); para este caso, las 16 señales moduladas se encuentran exactamente en sus posiciones ideales en el plano complejo. Para un mayor detalle, en el apéndice B, tabla B.5 se observan los valores de salida de la parte del codificador.

En la figura del centro, el diagrama muestra la misma señal modulada de la izquierda, pero con un nivel de ruido añadido de 10 dB. El ruido introduce perturbaciones causando que los símbolos se dispersen alrededor de sus posiciones ideales causando en esta ocasión interferencia entre los símbolos. Por lo tanto, la probabilidad de error se incrementará al momento de reconstruir los símbolos originales.

Por ultimo, la figura de la derecha muestra la constelación que corresponde a un SNR de 20 dB. El nivel de ruido es menor que en el caso de 10 dB; resultando en una menor dispersión de cada uno de los 16 símbolos. Aun con la presencia de ruido, para este nivel de SNR, se puede notar que no existe interferencia entre los símbolos teniendo como consecuencia una probabilidad de error más baja que el caso anterior.

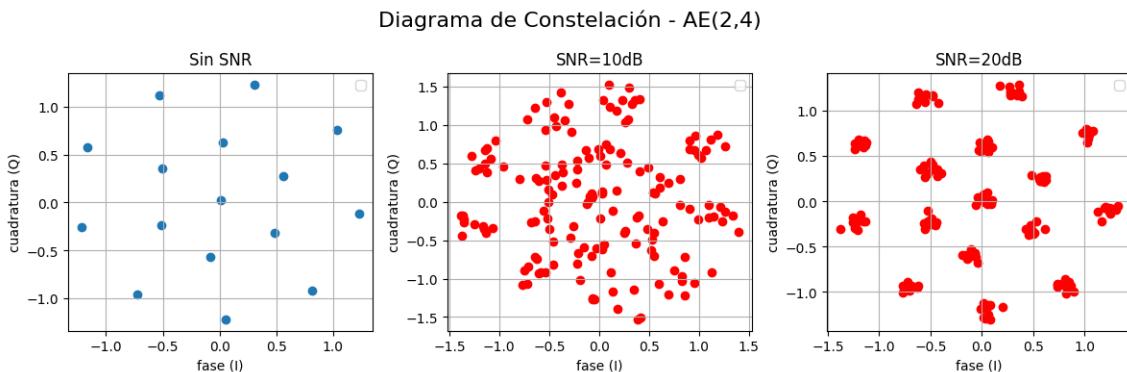


Figura 5.55: Diagrama de constelación para el auto codificador (2,4) normalizado en potencia.

Cuando se cuenta con número de canales mayores a 2 se recurre a reducir la dimensionalidad (proyección) a 2 dimensiones que representarán los valores de cada símbolo en el plano complejo. A continuación se presentan las proyecciones correspondientes a las tasas $R = 1$, $R = \frac{2}{3}$, y $R = \frac{1}{2}$; las cuales tiene más de 2 canales a la salida del codificador.

En las figuras 5.56, 5.57 y 5.58 se observa la reducción de dimensionalidad para 4, 6 y 8 canales respectivamente, en las subfiguras de la izquierda se utilizó el algoritmo de t-SNE en python en el cual se observa que para los tres casos no sigue una estructura parecida a la constelación de la figura 5.55, sin embargo logra tener una distancia prudente entre vecinos. Mientras que para las subfiguras de la derecha se utilizó el algoritmo de UMAP en python, el cual ofrece un resultado un poco similar al caso de la constelación observada en la figura 5.55 para los tres casos. Estas diferencias que se observan es debido a como cada algoritmo toma los datos para representarlos en la dimensión 2, cabe recalcar que los valores mostrados en las diferentes proyecciones no representan los valores originales en los cuales se encuentran los símbolos para dimensiones mayores a 2, simplemente es una forma de visualizar una representación adecuada para dimensiones en las cuales es imposible visualizar los puntos.

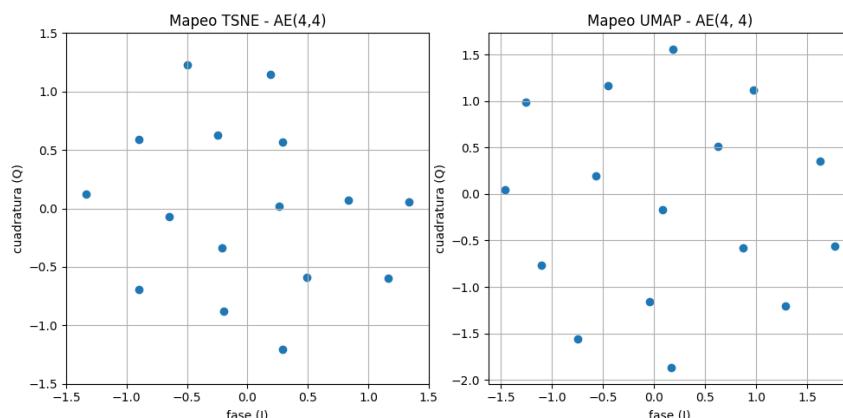


Figura 5.56: Constelaciones proyectadas para el auto codificador (4,4) normalizado en potencia.

5.8.3. Probabilidad de Error

En la figura 5.59 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificadores (n,4) y la modulación 16-Quadrature Amplitude Modulation (16-QAM) no codificada. La gráfica izquierda corresponde a una escala lineal, donde se aprecia como el BLER disminuye a medida que aumenta el E_b/N_0 para las diferentes tasas de código. Como se mencionó previamente la tasa $R = 2$ no es usada por lo que su BLER es el inferior de

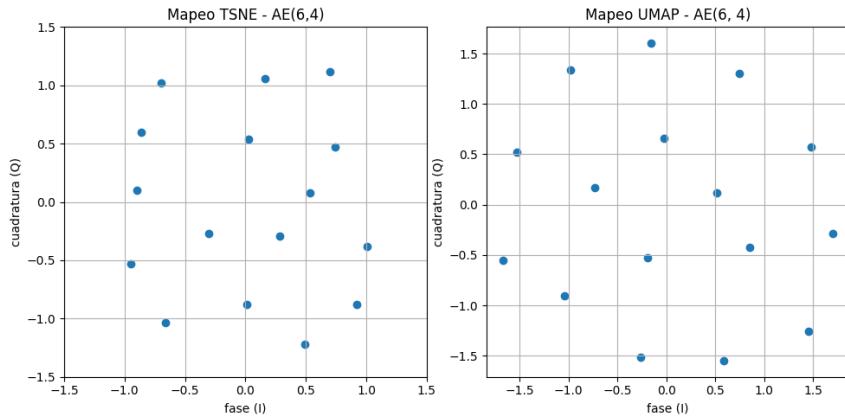


Figura 5.57: Constelaciones proyectadas para el auto codificador (6,4) normalizado en potencia.

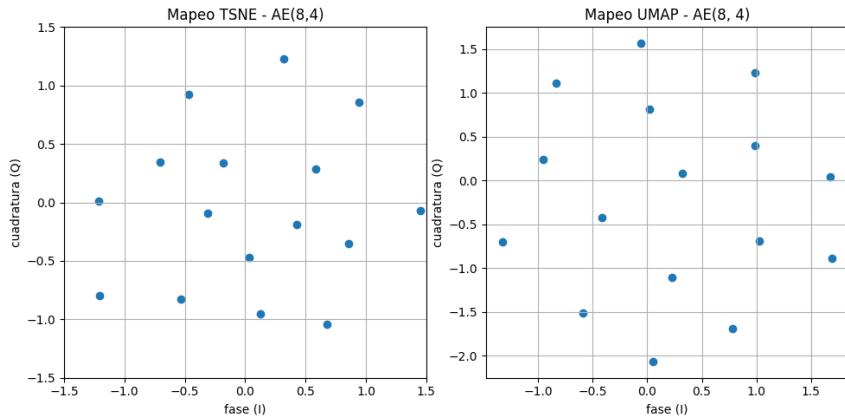


Figura 5.58: Constelaciones proyectadas para el auto codificador (8,4) normalizado en potencia.

todos los casos. Para una tasa $R = 1$ sin codificación (AE(4,4)), se aprecia el BLER mínimo requerido para un sistema confiable sin redundancia. Al variar la tasa de código a $R = \frac{2}{3}$ (AE(6,4)) y $R = 1/2$ (AE(8,4)) se puede observar que tiene un rendimiento cercano, siendo mejor la segunda tasa, esto debido a la redundancia en cada caso. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.1. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, muestra la relación BLER vs. E_b/N_0 en escala logarítmica, se compara el rendimiento de un sistema con auto codificador (n,4) frente a un sistema gls16-QAM no codificado. Inicialmente, para valores muy bajos de E_b/N_0 , el AE sin redundancia, muestra un desempeño ligeramente inferior que el sistema tradicional (16QAM(4,4)). Sin embargo, a partir de 2 dB, la curva del auto codificador (4,4), demuestra un mejor rendimiento. Esto se

debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información, superando al sistema gls16-QAM convencional, sin embargo para la tasa $R = 2$, el rendimiento resulta ser inferior que la modulación gls16-QAM sin codificación ya que como se mencionó en párrafos anteriores en vez de crear redundancia, se quita bits, por lo tanto, se trabaja con menor información lo que resulta en una mayor probabilidad de error.

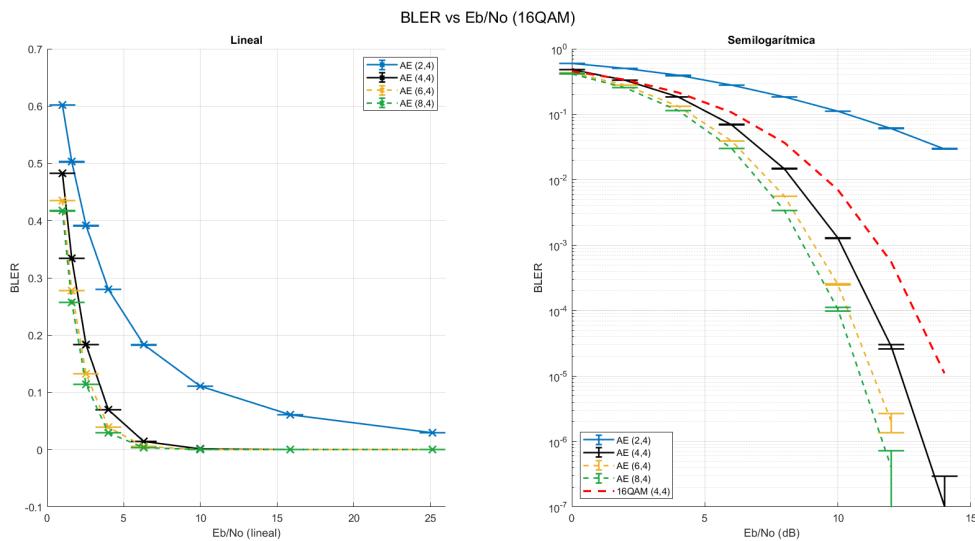


Figura 5.59: Gráfica del BLER para auto codificador (n,4) normalizado en potencia.

Las tasas que presentan redundancia $R = \frac{2}{3}$ y $R = \frac{1}{2}$ para valores de E_b/N_0 bajos hasta $\approx 2dB$ tienen un comportamiento muy similar, al sobrepasar ese valor el auto codificador que mejor rendimiento presenta es el de menor tasa ($R = \frac{1}{2}$), llegando a tener una diferencia de $\approx 0.5dB$ con respecto a la tasa $R = \frac{2}{3}$.

Sin embargo, con tasas de codificación más bajas, se necesita transmitir más bits para enviar la misma cantidad de información útil, lo que puede reducir la eficiencia espectral. Utilizar más ancho de banda para transmitir la misma cantidad de datos puede ser problemático en canales con ancho de banda limitado. Este análisis se amplía en la sección 5.12.

A continuación para analizar el comportamiento del BER para este auto codificador (n,4). En el apéndice D, tabla D.5 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales están en el rango de 10^{-4} hasta 10^{-8} , esto indica una dispersión mínima de los datos. En la figura 5.60 se muestran las curvas en escala lineal y logarítmica. Se nota que el auto codificador (2,4) presenta el inferior comportamiento (mayor bits erróneos) que todas las modulaciones incluyendo al sistema 16-QAM sin codificación, esto debido a la

falta de información que presenta este (de 4 bits se envían solo 2). Se hace notar que el auto codificador (4,4) presenta un inferior comportamiento en comparación con el sistema 16-QAM sin codificación, hasta $\approx 3dB$, en adelante el AE presenta una mejora. Esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información, superando al sistema gls16-QAM convencional. Para las otras tasas de codificación se observa un mejor rendimiento, teniendo el mejor comportamiento la tasa de $R = \frac{1}{2}$ (redundancia de 4 bits), llevando una ventaja de $\approx 1dB$ a comparación con el AE de tasa $R = \frac{2}{3}$ (redundancia de 2 bits). Al comparar estos resultados con la gráfica de BLER (figura 5.59) es visible que las curvas de BER empiezan en un valor más bajo indicando que la probabilidad de error a nivel de bit es menor, por debajo de 10^{-1} para las tasas más bajas, auto codificador (6,4) y (8,4), y un poco más arriba para el caso de las tasas más altas, auto codificador (2,4) y (4,4), incluyendo a la modulación sin codificación. Si el BER es menor al BLER se puede llegar a la conclusión que hay pocos bits erróneos en general, esos errores están distribuidos de manera que afectan a muchos bloques y en consecuencia el BLER aumenta.

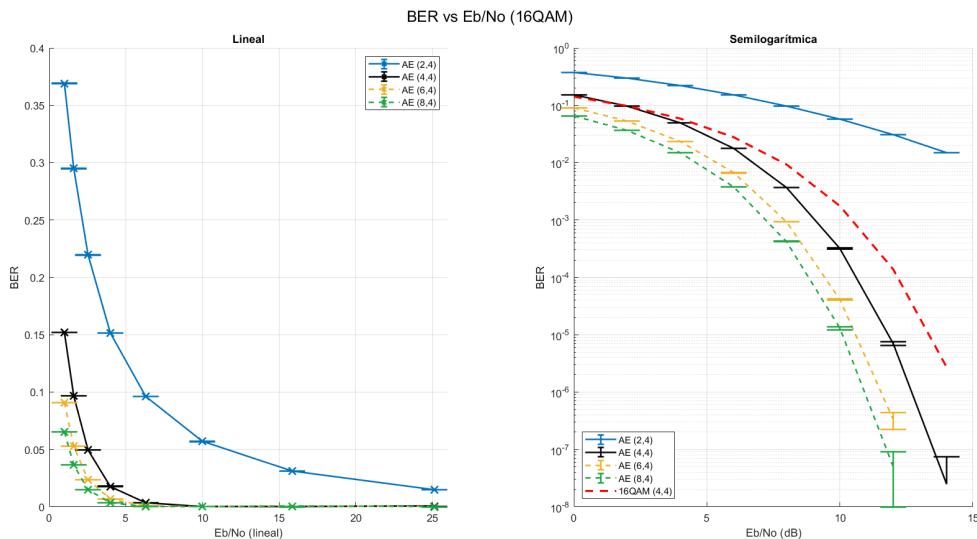


Figura 5.60: Gráfica del BER para auto codificador (n,4) normalizado en potencia.

5.9. 32-QAM - auto codificador (n,5)

5.9.1. Entrenamiento

El primer caso (figura 5.61) a considerar es el auto codificador (2,5) que genera una tasa de código $R = 2.5$. El valor de pérdida calculado teniendo en cuenta $M = 32$ símbolos o clases,

con probabilidad uniforme de $p_{ij} = 0.03125$:

$$L = - \sum_{i=1}^{32} (y_{ij} \ln(0.03125)) = 3.46 \quad (5.11)$$

El valor inicial de pérdida observado en la figura 5.61 es ≈ 2.1 al inicio del entrenamiento, decrece y se estabiliza ≈ 1.3 ; por otro lado, los valores de pérdida para la validación comienza en 1.1 y decrece a ≈ 0.5 luego de 18 épocas. Para la gráfica de precisión, empieza con un valor ≈ 0.3 y alcanza ≈ 0.55 en el entrenamiento; sin embargo, para el set de validación comienza en ≈ 0.85 y se estabiliza en ≈ 1 ; este comportamiento se debe a la tasa empleada que es mayor a 1 en la cual se está eliminando información para cada símbolo; es decir, la red neuronal no es capaz de ajustar los pesos para minimizar los errores. Esta tasa se considera únicamente para comparación y efectos de visualización para el diagrama de constelación.

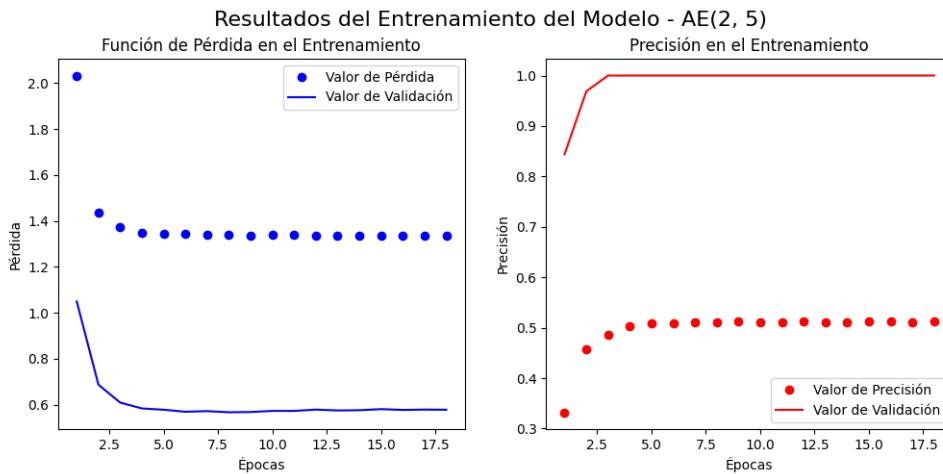


Figura 5.61: Resultados del entrenamiento para el auto codificador (2,5) normalizado en potencia.

El segundo caso para análisis es para el auto codificador (5,5), que produce una tasa de código $R = 1$, es decir, no se agrega redundancia. En la figura 5.62 el valor inicial de pérdida, en el entrenamiento, es ≈ 1.05 y decrece hasta 0.4 en 26 épocas; mientras, el valor de pérdida para la validación empieza en ≈ 0.05 y decrece rápidamente hasta ≈ 0 . En la gráfica de la precisión, en el entrenamiento, se observa que inicia con un valor de 0.74 y alcanza ≈ 0.86 . Para la validación en un inicio alcanza ≈ 1 , este comportamiento se debe a el sistema emplea la misma cantidad de bits y canales para transmitir.

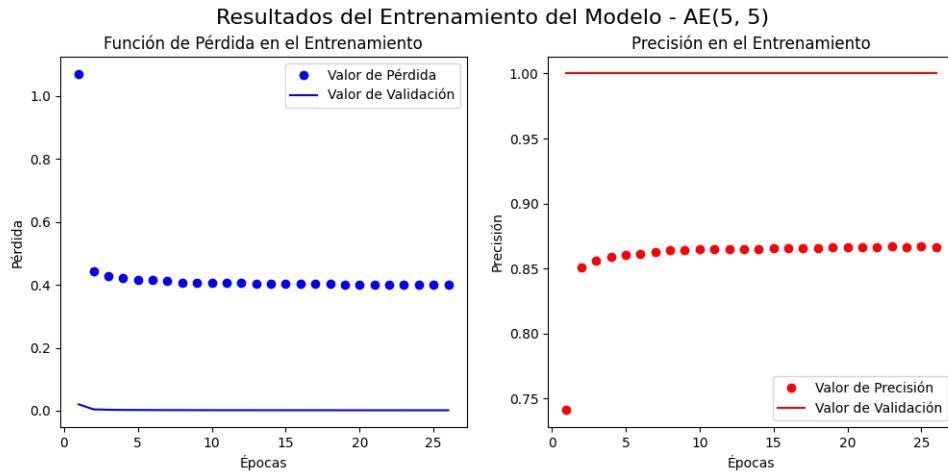


Figura 5.62: Resultados del entrenamiento para el auto codificador (5,5) normalizado en potencia.

El tercer caso para análisis es el auto codificador (6,5), que produce una tasa de código $R = \frac{5}{6}$, es decir, se agrega redundancia del 16 %. En la figura 5.63 el valor inicial de pérdida, en el entrenamiento, es ≈ 1 y decrece hasta 0.3 (10 % más que la tasa $R = 1$) en 30 épocas; mientras, que el valor de pérdida para la validación empieza en ≈ 0 . En la gráfica de la precisión, para el entrenamiento, el valor inicial es 0.75 y alcanza ≈ 0.9 (4 % más que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, como resultado de emplear un canal adicional que el caso anterior, permitiendo una mejor generalización de los datos.

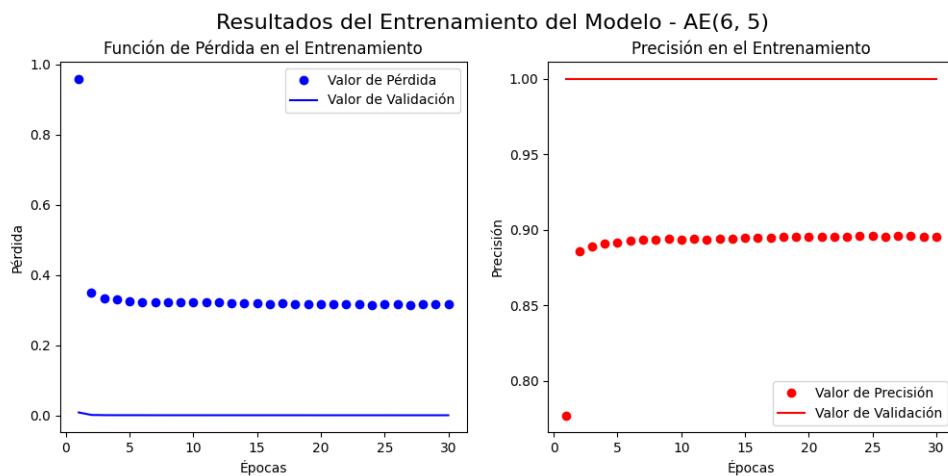


Figura 5.63: Resultados del entrenamiento para el auto codificador (6,5) normalizado en potencia.

El cuarto caso, es para el auto codificador (10,5), que produce una tasa de código $R = \frac{1}{2}$, es decir, se agrega redundancia del 50 %. En la figura 5.64 el valor inicial de pérdida, en el

entrenamiento, es ≈ 0.9 y decrece hasta 0.2 en 18 épocas (se reduce a la mitad de $R = 1$); mientras, que el valor de pérdida para la validación empieza relativamente en ≈ 0 , esto por el efecto de la redundancia. En la gráfica de la precisión, en el entrenamiento, el valor inicial es 0.825 y alcanza ≈ 0.94 ($\approx 8\%$ más que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, como resultado de emplear mas canales adicionales que los casos anteriores.

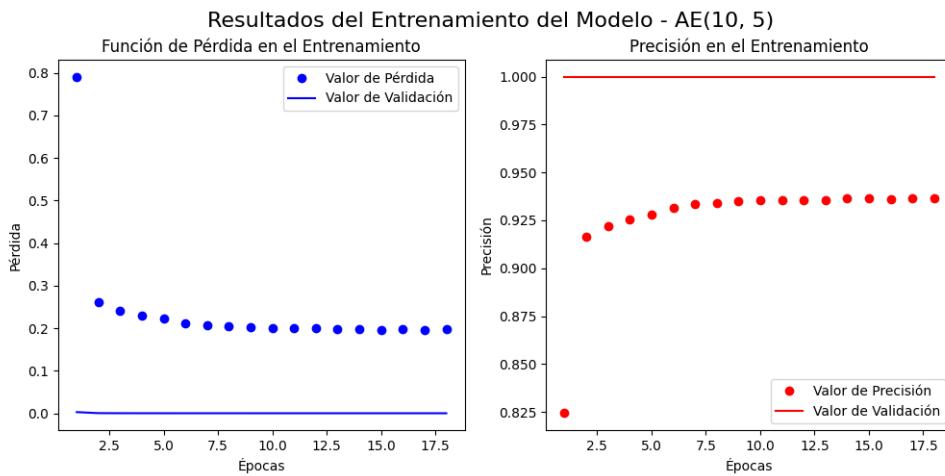


Figura 5.64: Resultados del entrenamiento para el auto codificador (10,5) normalizado en potencia.

En conclusión, los mejores entrenamientos que se obtuvo son con las tasas $R = \frac{5}{6}$ y $R = \frac{1}{2}$ ya que permite agregar bits de redundancia a la información permitiendo a la red decodificador mejorar los datos; sin embargo, el aumento de canales genera la necesidad de una mayor número de épocas para un mejor entrenamiento, lo cual implica la necesidad de mayor tiempo y recursos para el entrenamiento tal como se muestra en la tabla 5.9.

5.9.2. Diagrama de Constelación

En la figura 5.65 se observan que el diagrama de constelación para el auto codificador (2,5) converge en uno de 32-Quadrature Amplitude Modulation (32-QAM). La figura de la izquierda muestra una constelación sin la presencia de ruido (salida del codificador); para este caso, las 32 señales moduladas se encuentran exactamente en sus posiciones ideales en el plano complejo y distribuidas uniformemente. Para un mayor detalle, en el apéndice B, tabla B.6 se observan los valores de salida de la parte del codificador.

En las otras dos constelaciones se introduce ruido, es decir, es la representación de los símbolos que entrarán en el decodificador. En la figura del centro, el diagrama muestra la misma

señal modulada de la izquierda, pero con un nivel de señal a ruido correspondiente a 10 dB, el ruido introduce perturbaciones en la posición de cada uno de los 32 símbolos, lo que causa que se produzca la interferencia entre los símbolos tal como se muestra en la constelación; consiguiendo aumentar la probabilidad de error al momento de querer reconstruir los mismos.

Por último, la figura de la derecha muestra la constelación que corresponde a un SNR de 20 dB. El nivel de ruido es menor que en el caso de 10 dB; resultando en una menor dispersión de cada uno de los 32 símbolos alrededor de sus posiciones ideales, para este nivel de SNR se observa que la probabilidad de una interferencia entre los símbolos es casi nula, logrando una baja probabilidad de error. Esto contrasta con la figura 5.37, ya que en este caso la distribución de los símbolos es de una manera más amplia debido al tipo de normalización implementada (potencia), ocasionando así que no exista solapamiento entre símbolos.

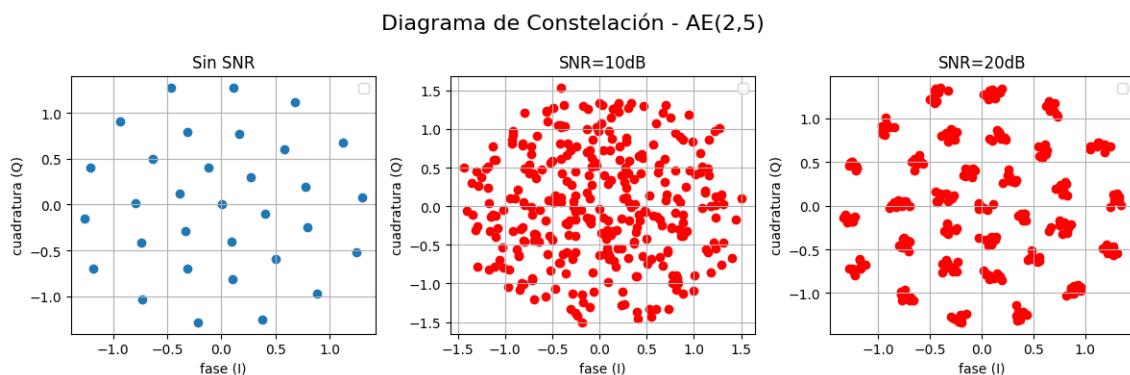


Figura 5.65: Diagrama de constelación para el auto codificador (2,5) normalizado en potencia.

Cuando se cuenta con número de canales mayores a 2 se recurre a reducir la dimensionalidad (proyección) a 2 dimensiones que representarán los valores de cada símbolo en el plano complejo. A continuación se presentan las proyecciones correspondientes a las tasas $R = 1$, $R = \frac{5}{6}$, y $R = \frac{1}{2}$; las cuales tiene más de 2 canales a la salida del codificador.

En las figuras 5.66, 5.67 y 5.68 se observa la reducción de dimensionalidad para 5, 6 y 10 canales respectivamente, en las subfiguras de la izquierda se utilizó el algoritmo de t-SNE en python en el cual se observa que no sigue una estructura parecida a la constelación de la figura 5.65 salvo el caso del auto codificador (6,5), el cual pretende seguir la forma pero la escala de valores se encuentra reducida a la mitad. Mientras que para las subfiguras de la derecha se utilizó el algoritmo de UMAP en python, el cual ofrece un resultado un poco mejor a las anteriores pero de igual manera con una brecha grande de similitud con la figura 5.65. Estas diferencias que se observan es debido a como cada algoritmo toma los datos para representarlos en la dimensión 2, cabe recalcar que los valores mostrados en las diferentes

proyecciones no representan los valores originales en los cuales se encuentran los símbolos para dimensiones mayores a 2, simplemente es una forma de visualizar una representación adecuada para dimensiones en las cuales es imposible visualizar los símbolos.

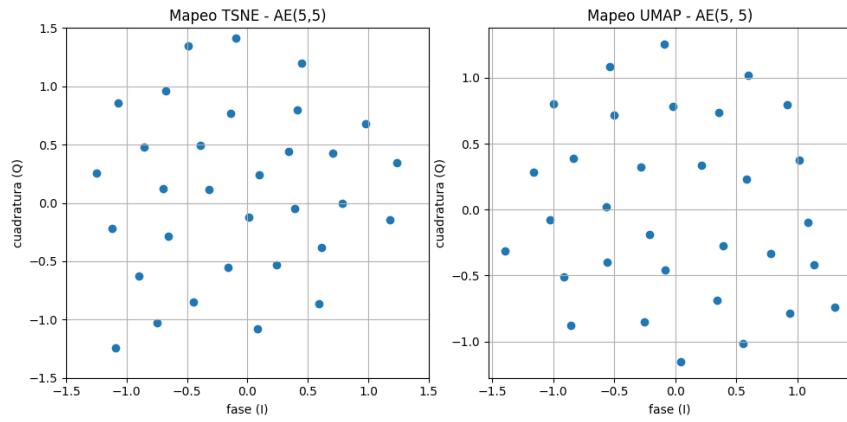


Figura 5.66: Constelaciones proyectadas para el auto codificador (5,5) normalizado en potencia.

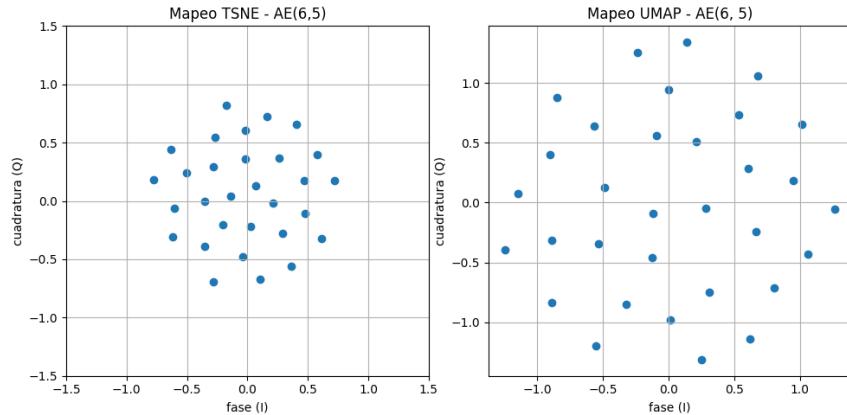


Figura 5.67: Constelaciones proyectadas para el auto codificador (6,5) normalizado en potencia.

5.9.3. Probabilidad de Error

En la figura 5.41 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (n,5) y la modulación 32-QAM sin codificación. La gráfica izquierda corresponde a una escala lineal, donde se aprecia cómo el BLER disminuye a medida que aumenta el E_b/N_0 para las diferentes tasas de código. Como se mencionó previamente la tasa $R = \frac{5}{2}$ no es usada por lo que su BLER es el inferior de todos los casos. Las tasas de código

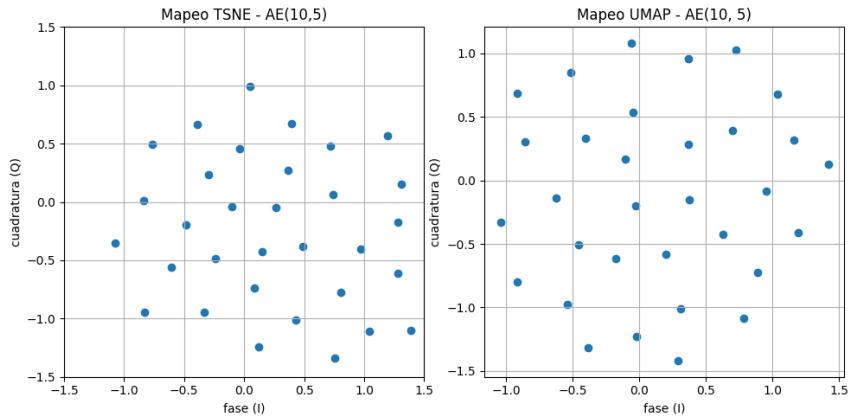


Figura 5.68: Constelaciones proyectadas para el auto codificador (10,5) normalizado en potencia.

$R = \frac{5}{6}$ (AE(6,5)) y $R = 1/2$ (AE(10,5)) se puede observar que tienen un mejor rendimiento que la tasa $R = 1$, siendo la mejor opción la segunda tasa ya que posee mayor redundancia (auto codificador (10,5), se agrega 5 bits de redundancia). Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-6} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.6. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, que muestra la relación BLER vs. E_b/N_0 en escala logarítmica, se compara el rendimiento de un sistema con auto codificador (n,5) frente a un sistema 32-QAM sin codificación. Inicialmente, para valores muy bajos de E_b/N_0 , el AE con $R \leq 1$ muestra un desempeño mejor que el sistema tradicional (32QAM(5,5)); sugiriendo la capacidad de corrección de errores en valores bajos de E_b/N_0 . La curva del auto codificador (5,5), demuestra un mejor rendimiento que el sistema base (32QAM(5,5)); esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información, superando al sistema 32-QAM estándar. Mientras que para la tasa $R = \frac{5}{2}$ AE(2,5)), el rendimiento es mucho inferior que para la modulación 32-QAM debido a que en esta ocasión no se está incorporando redundancia, si no al contrario quitando información.

Aunque para $0dB$ el comportamiento es casi el mismo para el auto codificador con $R \leq 1$, a medida que va creciendo los valores de E_b/N_0 las tasas más pequeñas del auto codificador tienen un mejor resultado, siendo el codificador (10,5) el que mejor rendimiento presenta llevando una ventaja de $\approx 1dB$ en comparación con el auto codificador (6,5); esto se debe a la mayor redundancia que posee el primero (4 bits más de redundancia).

Sin embargo, con tasas de codificación más bajas, se necesita transmitir más bits para enviar

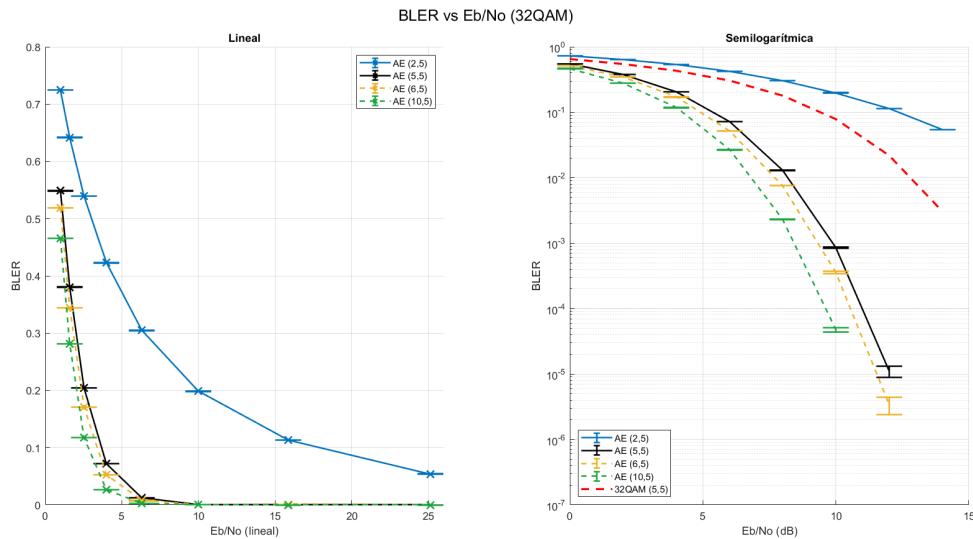


Figura 5.69: Gráfica del BLER para auto codificador (n,5) normalizado en potencia.

la misma cantidad de información útil, lo que puede reducir la eficiencia espectral. Utilizar más ancho de banda para transmitir la misma cantidad de datos puede ser problemático en canales con ancho de banda limitado. Este análisis se amplía en la sección [5.12](#).

A continuación para analizar el comportamiento del BER para este auto codificador (n,5). En el apéndice D, tabla D.6 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales están en el rango de 10^{-4} hasta 10^{-7} , esto indica una dispersión mínima de los datos. En la figura 5.70 se muestran las curvas en escala lineal y logarítmica. Se nota que el auto codificador(2,5) presenta el inferior comportamiento (mayor bits erróneos) que todas las modulaciones incluyendo al sistema 32-QAM sin codificación, esto debido a la falta de información que presenta este (de 5 bits se envían solo 2). Se hace notar que el auto codificador (5,5) tiene una mayor probabilidad de error que el sistema 32-QAM sin codificación hasta $\approx 3dB$ (punto de intersección), en adelante el AE presenta una mejora; esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información. Para las otras tasas de codificación se observa un mejor rendimiento, teniendo el mejor comportamiento la tasa de $R = \frac{1}{2}$ (redundancia de 5 bits), llevando una ventaja de $\approx 1dB$ a comparación con el AE de tasa $R = \frac{5}{6}$ (redundancia de 1 bits). Al comparar estos resultados con la gráfica de BLER (figura 5.69) es visible que las curvas de BER empiezan en un valor más bajo indicando que la probabilidad de error a nivel de bit es menor, por debajo de 10^{-1} para la tasas más baja, auto codificador (10,5), y un poco más arriba para el caso de las tasas más altas, auto codificador (2,5), (5,5) (6,5), incluyendo a la modulación sin codificación. Si el BER es menor al BLER se puede llegar a la conclusión que hay pocos bits erróneos

en general, esos errores están distribuidos de manera que afectan a muchos bloques y en consecuencia el BLER aumenta.

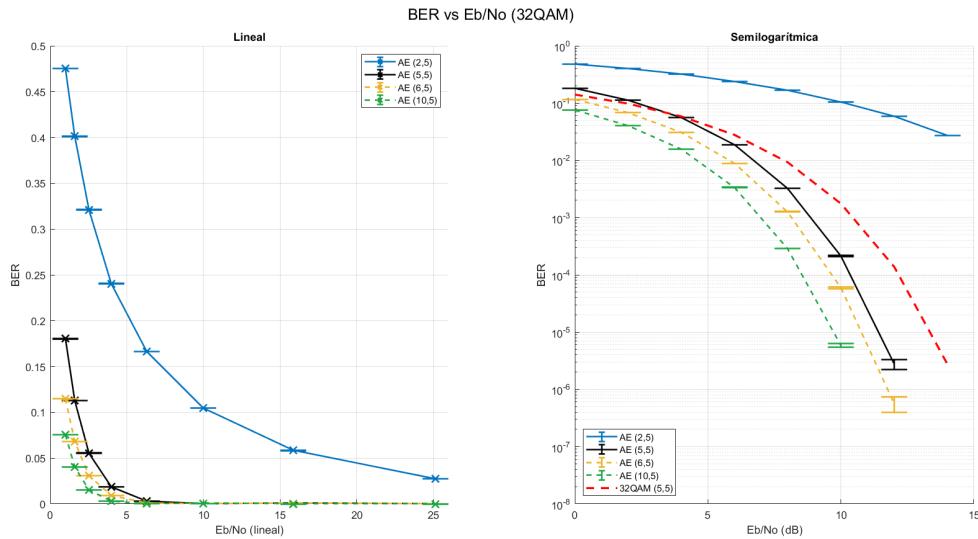


Figura 5.70: Gráfica del BER para auto codificador (n,5) normalizado en potencia.

5.10. 64-QAM - auto codificador (n,6)

5.10.1. Entrenamiento

El primer caso a considerar es el auto codificador (2,6) el cual genera una tasa de código $R = 3$, tasa que no se usa ya que al ser mayor a 1 en vez de agregar redundancia de bits a los símbolos, se estarían quitando (de los 6 bits se enviarían solo 2 bits). El valor de pérdida calculado teniendo en cuenta $M = 64$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.015625$:

$$L = - \sum_{i=1}^{64} (y_{ij} \ln(0.015625)) = 4.158 \quad (5.12)$$

El valor inicial de pérdida observado en la figura 5.71 es ≈ 2.25 al inicio del entrenamiento (casi la mitad del valor calculado), decrece y se estabiliza ≈ 0.9 ; por otro lado, los valores de pérdida para la validación comienza en 1.05 y decrece a ≈ 0.20 luego de 17 épocas. Para la gráfica de precisión, empieza con un valor ≈ 0.35 y alcanza ≈ 0.68 en el entrenamiento; sin embargo, para el set de validación comienza en ≈ 0.95 y se estabiliza en ≈ 1 ; este comportamiento se debe a la tasa empleada que es mayor a 1 en la cual se está eliminando información para

cada símbolo; es decir, la red no es capaz de ajustar los pesos para permitir minimizar los errores. Esta tasa se considera únicamente para comparación y efectos de visualización para el diagrama de constelación.

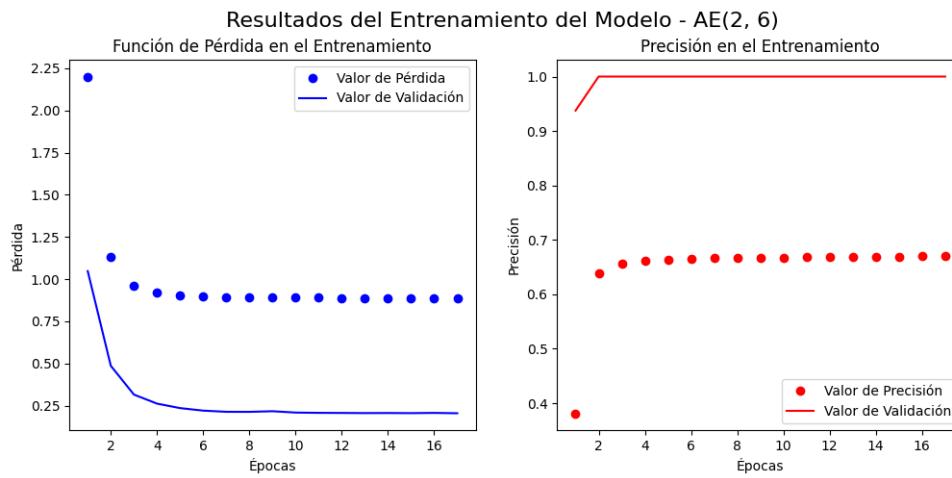


Figura 5.71: Resultados del entrenamiento para el auto codificador (2,6) normalizado en potencia.

El segundo caso de análisis es para el auto codificador (6,6), que produce una tasa de código $R = 1$, es decir, no se agrega redundancia. En la figura 5.72 el valor inicial de pérdida, en el entrenamiento, es ≈ 0.7 y decrece hasta 0.03 en 21 épocas; mientras, el valor de pérdida para la validación empieza en ≈ 0 . En la gráfica de la precisión, en el entrenamiento, se observa que inicia con un valor de 0.92 y alcanza ≈ 1 . Para la validación en un inicio alcanza ≈ 1 , este comportamiento se debe a que a pesar de que no exista redundancia de bits el ajuste de los pesos ayuda a que el modelo realice de mejor manera la generalización y este listo para cualquier conjunto de datos, evitando el sobreajuste.

El tercer caso para análisis es el auto codificador (6,5), que produce una tasa de código $R = \frac{2}{3}$, es decir, se agrega redundancia del 33 %. En la figura 5.73 el valor inicial de pérdida, en el entrenamiento, es ≈ 0.55 y decrece hasta 0 (15 % menos pérdida al iniciar que la tasa $R = 1$, pero mismo valor al estabilizarse) en 25 épocas; mientras, que el valor de pérdida para la validación empieza en ≈ 0 . En la gráfica de la precisión, para el entrenamiento, el valor inicial es 0.94 y alcanza ≈ 1 (2 % mayor precisión que con la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, como resultado de emplear 3 canales adicionales (3 bits de redundancia) que el caso anterior, permitiendo una mejor generalización de los datos.

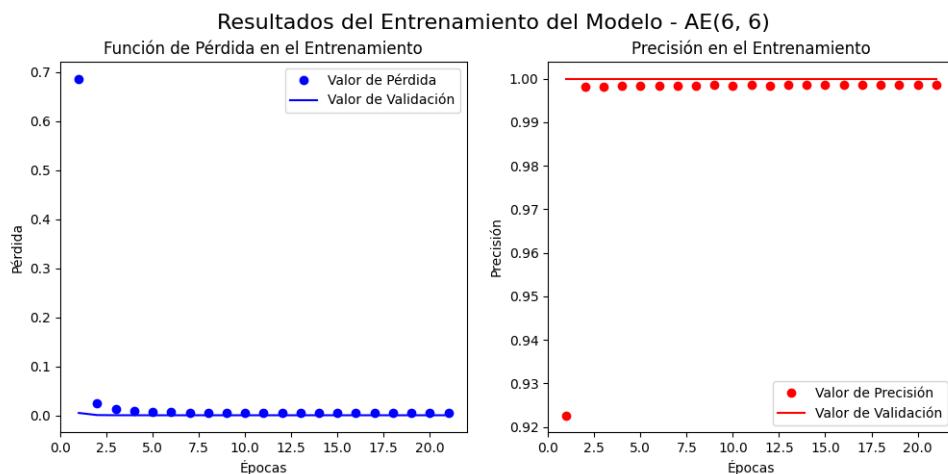


Figura 5.72: Resultados del entrenamiento para el auto codificador (6,6) normalizado en potencia.

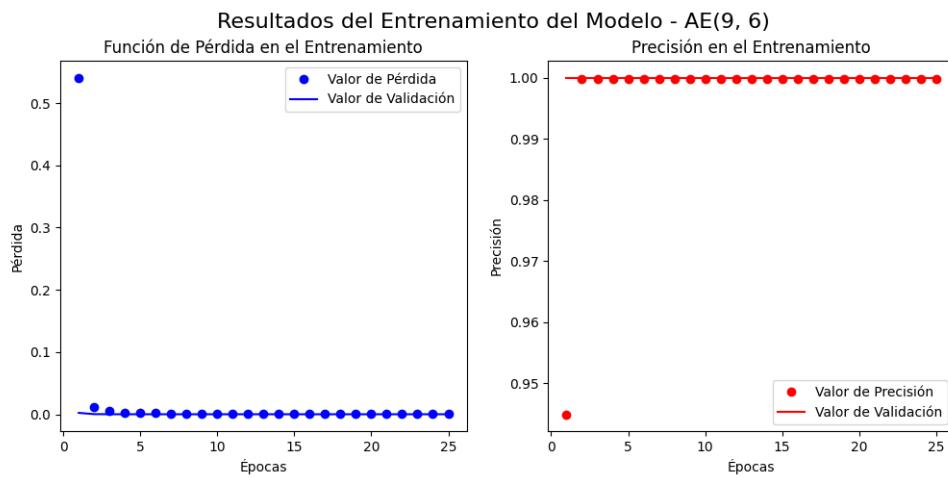


Figura 5.73: Resultados del entrenamiento para el auto codificador (9,6) normalizado en potencia.

El cuarto caso, es para el auto codificador (12,6), que produce una tasa de código $R = \frac{1}{2}$, es decir, se agrega redundancia del 50 %. En la figura 5.74 el valor inicial de pérdida, en el entrenamiento, es ≈ 0.5 y decrece hasta 0 en 18 épocas (pérdida reducida en un 20 % en comparación a tasa $R = 1$ al iniciar el entrenamiento); mientras, que el valor de pérdida para la validación empieza relativamente en ≈ 0 . En la gráfica de la precisión, para los datos de entrenamiento, el valor inicial es 0.96 y alcanza ≈ 1 (≈ 4 % más que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, esto se obtiene como resultado de emplear más canales adicionales que los casos anteriores.

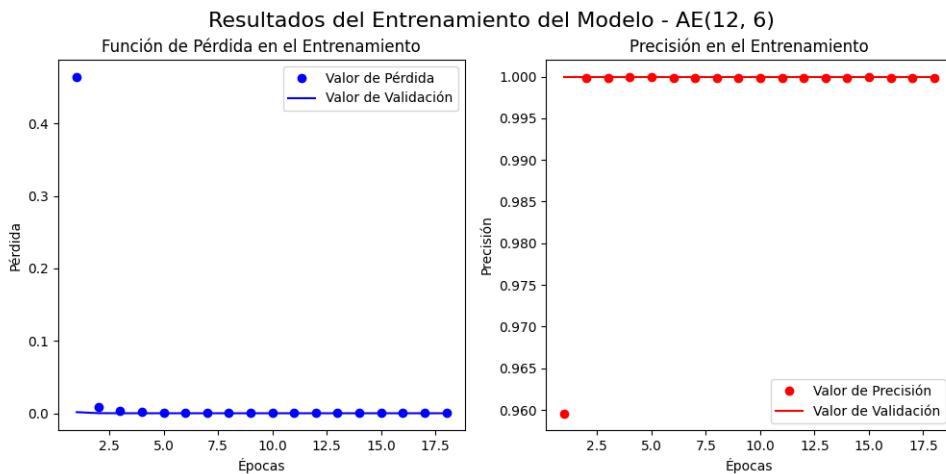


Figura 5.74: Resultados del entrenamiento para el auto codificador (12,6) normalizado en potencia.

En conclusión, los mejores entrenamientos que se obtuvieron son con todas las tasas excepto con $R = 3$ debido a que es mayor a 1 y el envío de los símbolos tiene pérdidas. En todas las demás tasas se logró llegar al mínimo en la pérdida y el máximo en la precisión, ambos parámetros tanto para datos de entrenamiento como validación. Sin embargo, como se ha mencionado en las codificaciones anteriores el aumento de canales genera la necesidad de un mayor tiempo y recursos para el entrenamiento tal como se muestra en la tabla 5.9.

5.10.2. Diagrama de Constelación

En la figura 5.75 se observa el diagrama de constelación para el auto codificador (2,6) el cual converge en uno de 64-Quadrature Amplitude Modulation (64-QAM), al no ser una constelación circular. La figura de la izquierda muestra una constelación sin la presencia de ruido (salida del codificador); para este caso, los 64 símbolos se encuentran exactamente en sus posiciones ideales en el plano complejo y distribuidos uniformemente. Para un mayor detalle, en el apéndice B, tabla B.7 se observan los valores de salida de la parte del codificador.

En las otras dos constelaciones se introduce ruido, es decir, es la representación de los símbolos que entrarán en el decodificador. En la figura del centro, el diagrama muestra la misma señal modulada de la izquierda, pero con un nivel de señal a ruido correspondiente a 20 dB, el ruido introduce perturbaciones en la posición de cada uno de los 64 símbolos, lo que causa que se produzca la interferencia entre los mismos tal como se muestra en la constelación; consiguiendo aumentar la probabilidad de error al momento de querer reconstruirlos.

Por último, la figura de la derecha muestra la constelación que corresponde a un SNR de 30 dB. El nivel de ruido es menor que en el caso de 20 dB; resultando en una menor dispersión de cada uno de los 64 símbolos alrededor de sus posiciones ideales, para este nivel de SNR se observa que la probabilidad de una interferencia entre los símbolos es casi nula para los que se encuentran en las afueras y una probabilidad considerable para los símbolos que se encuentran rodeando el origen. Se nota además que para este auto codificador como para el siguiente, el valor de SNR utilizado en las constelaciones es mayor debido al incremento de símbolos y por ende a una mejor apreciación y distinción del efecto del ruido en estos.

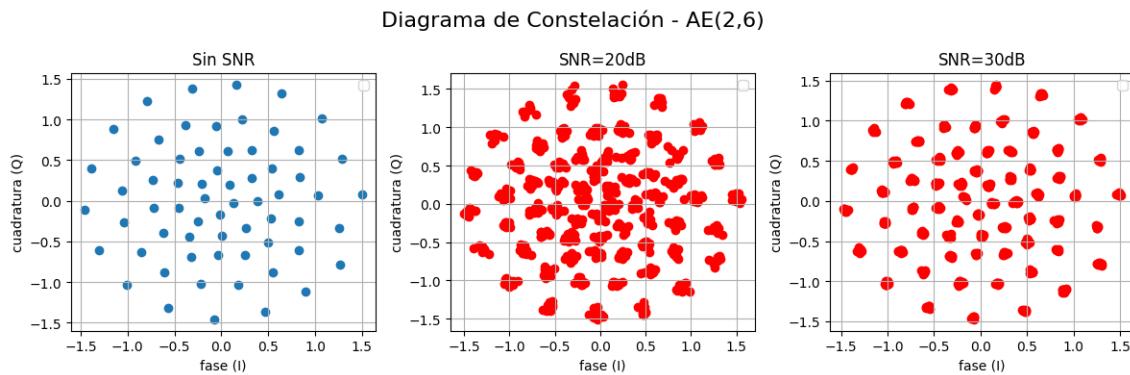


Figura 5.75: Diagrama de constelación para el auto codificador (2,6) normalizado en potencia.

Cuando se cuenta con número de canales mayores a 2 se recurre a reducir la dimensionalidad (proyección) a 2 dimensiones que representarán los valores de cada símbolo en el plano complejo. A continuación se presentan las proyecciones correspondientes a las tasas $R = 1$, $R = \frac{2}{3}$, y $R = \frac{1}{2}$; las cuales tiene más de 2 canales a la salida del codificador.

En las figuras 5.76, 5.77 y 5.78 se observa la reducción de dimensionalidad para 6, 9 y 12 canales respectivamente, en las subfiguras de la izquierda se utilizó el algoritmo de t-SNE en python en el cual se observa que al contrario de las modulaciones anteriores este sigue una estructura parecida a la de la figura 5.75. Mientras que para las subfiguras de la derecha se utilizó el algoritmo de UMAP en python, el cual ofrece un resultado no tan exitoso como para los casos anteriores, aunque los diagramas pretendan seguir la estructura modelo no son tan certeros como los mapeados utilizando t-SNE. Estas diferencias que se observan es debido a como cada algoritmo toma los datos para representarlos en la dimensión 2, cabe recalcar que los valores mostrados en las diferentes proyecciones no representan los valores originales en los cuales se encuentras los símbolos para dimensiones mayores a 2, simplemente es una forma de visualizar una representación adecuada para dimensiones en las cuales es imposible visualizar los símbolos.

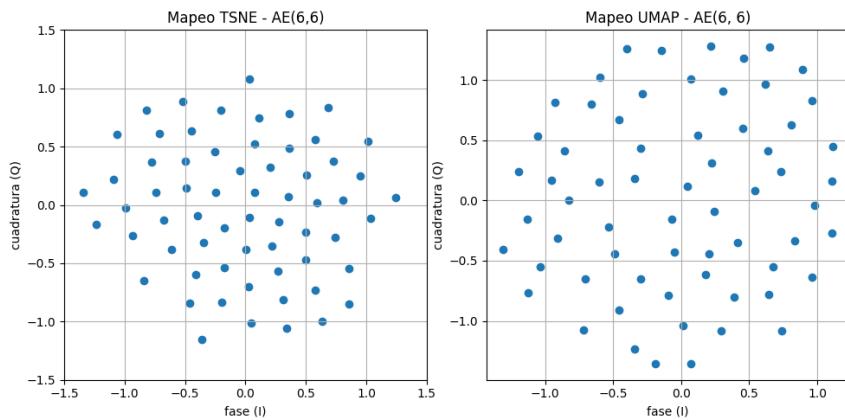


Figura 5.76: Constelaciones proyectadas para el auto codificador (6,6) normalizado en potencia.

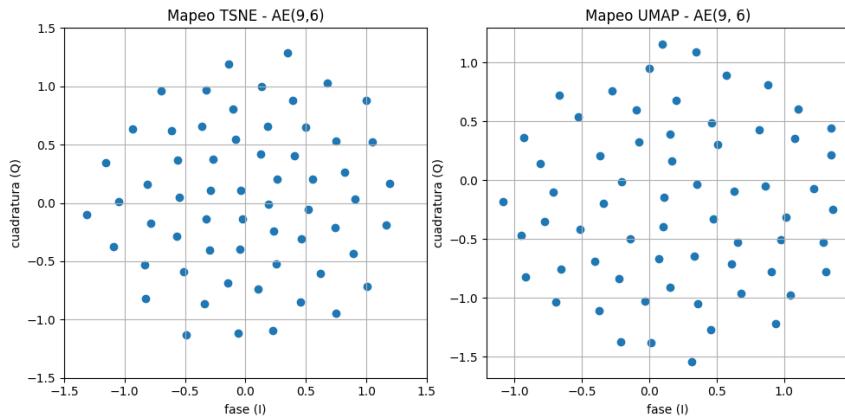


Figura 5.77: Constelaciones proyectadas para el auto codificador (9,6) normalizado en potencia.

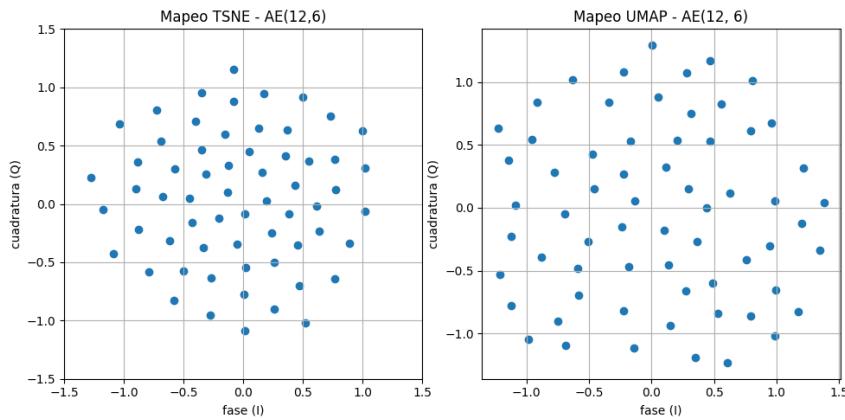


Figura 5.78: Constelaciones proyectadas para el auto codificador (12,6) normalizado en potencia.

5.10.3. Probabilidad de Error

En la figura 5.79 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (n,6) y la modulación 64-QAM sin codificación. La gráfica izquierda corresponde a una escala lineal, donde se aprecia como el BLER disminuye a medida que aumenta el E_b/N_0 para las diferentes tasas de código. Como se mencionó previamente la tasa $R = 3$ (se envían 3 bits de los 6 bits codificados) no es usada por lo que su BLER es el inferior de todos los casos. Al variar la tasa de código a $R = \frac{2}{3}$ (AE(9,6)) y $R = 1/2$ (AE(12,6)) se puede observar que tienen un mejor rendimiento que $R = 1$, siendo la mejor opción la segunda tasa, a pesar de que difiere de la tasa $R = \frac{2}{3}$ por un valor muy pequeño. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.7. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, que muestra la relación BLER vs. E_b/N_0 en escala logarítmica, se compara el rendimiento de un sistema con auto codificador (n,6) frente a un sistema 64-QAM sin codificación. Inicialmente, para valores muy bajos de E_b/N_0 , el AE con $R \leq 1$ muestra un desempeño mejor que el sistema tradicional (64QAM (6,6)); sugiriendo la capacidad de corrección de errores en valores bajos de E_b/N_0 . La curva del auto codificador (6,6), demuestra un mejor rendimiento que el sistema sin codificación; esto se debe a que, aunque el sistema no tiene codificación de bloque, ha aprendido a decodificar de manera más eficiente la información, superando al sistema 64-QAM estándar. Mientras que para la tasa $R = 3$ (AE(2,6)), el rendimiento es mucho inferior que para la modulación 64-QAM debido a que en esta ocasión no se está incorporando redundancia, si no al contrario quitando información.

Aunque para $0dB$ el comportamiento es casi el mismo para el auto codificador con $R \leq 1$, a medida que va creciendo los valores de E_b/N_0 las tasas más pequeñas del auto codificador tienen un mejor resultado, siendo el codificador (12,6) el que mejor rendimiento presenta llevando una ventaja de $\approx 0.25dB$ en comparación con el auto codificador (9,6); esto se debe a la mayor redundancia que posee el primero (3 bits más de redundancia).

Sin embargo, con tasas de codificación más bajas, se necesita transmitir más bits para enviar la misma cantidad de información útil, lo que puede reducir la eficiencia espectral. Utilizar más ancho de banda para transmitir la misma cantidad de datos puede ser problemático en canales con ancho de banda limitado. Este análisis se amplía en la sección 5.12.

A continuación para analizar el comportamiento del BER para este auto codificador (n,6). En

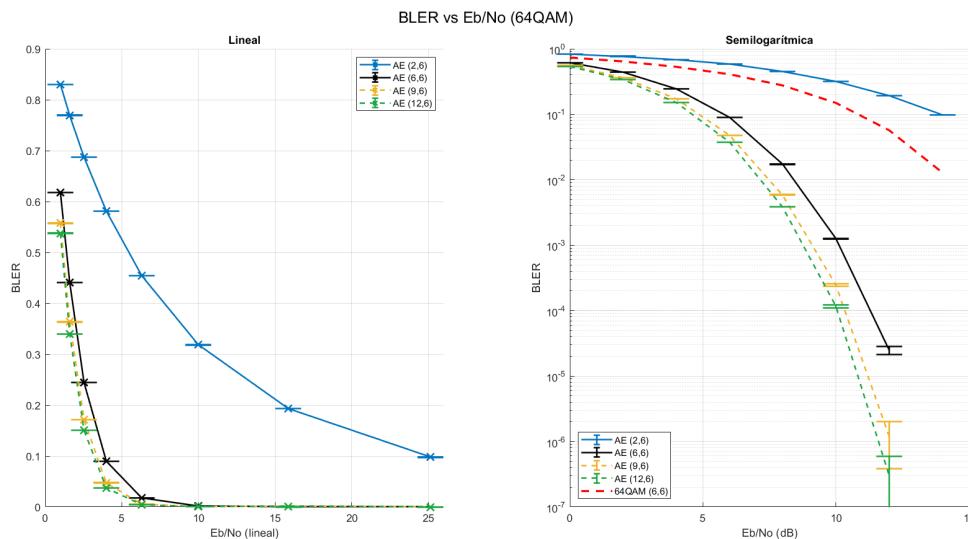


Figura 5.79: Gráfica del BLER para auto codificador (n,6) normalizado en potencia.

el apéndice D, tabla D.7 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales están en el rango de 10^{-4} hasta 10^{-8} , esto indica una dispersión mínima de los datos. En la figura 5.80 se muestran las curvas en escala lineal y logarítmica. Se nota que el auto codificador (2,6) presenta el inferior comportamiento (mayor bits erróneos) que todas las modulaciones incluyendo al sistema 64-QAM sin codificación, esto debido a la falta de información que presenta este (de 6 bits se envían solo 2). Se hace notar que el auto codificador (6,6) tiene una mayor probabilidad de error que el sistema 64-QAM sin codificación hasta $\approx 1\text{dB}$ (punto de intersección), en adelante el AE presenta una mejora; esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información. Para las otras tasas de codificación se observa un mejor rendimiento, teniendo el mejor comportamiento la tasa de $R = \frac{1}{2}$ (redundancia de 6 bits), llevando una ventaja de $\approx 0.5\text{dB}$ a comparación con el AE de tasa $R = \frac{2}{3}$ (redundancia de 3 bits). Al comparar estos resultados con la gráfica de BLER (figura 5.79) es visible que las curvas de BER empiezan en un valor más bajo indicando que la probabilidad de error a nivel de bit es menor, por debajo de 10^{-1} para la tasas más baja, auto codificador (12,6), y un poco más arriba para el caso de las tasas más altas, auto codificador (2,6), (6,6) (9,6), incluyendo a la modulación sin codificación. Si el BER es menor al BLER se puede llegar a la conclusión que hay pocos bits erróneos en general, esos errores están distribuidos de manera que afectan a muchos bloques y en consecuencia el BLER aumenta.

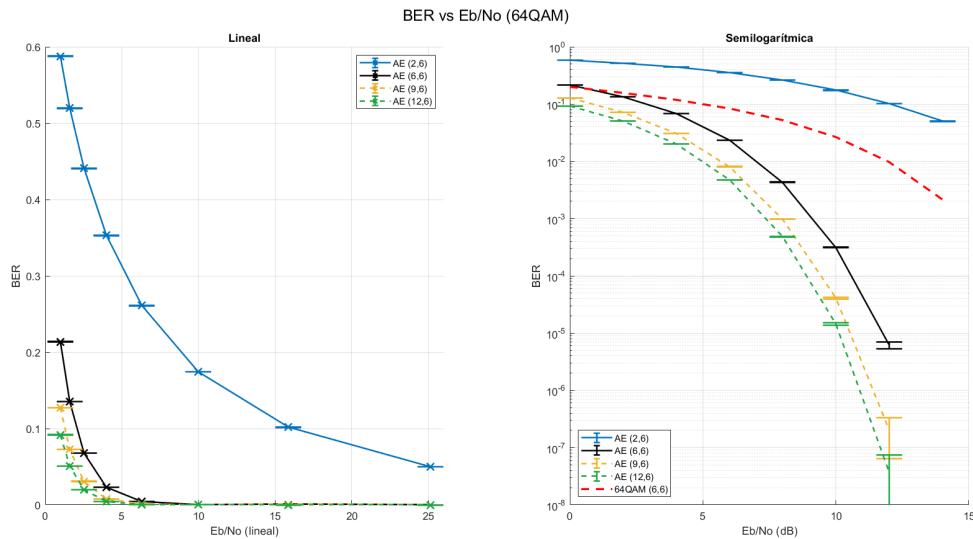


Figura 5.80: Gráfica del BER para auto codificador (n,6) normalizado en potencia.

5.11. 128-QAM - auto codificador (n,7)

5.11.1. Entrenamiento

El primer caso a considerar es el auto codificador (2,7) el cual genera una tasa de código $R = \frac{7}{2} = 3.5$, tasa que no se usa ya que al ser mayor a 1 en vez de agregar redundancia de bits a los símbolos, se estarían quitando (de los 7 bits se enviarían solo 2 bits). El valor de pérdida calculado teniendo en cuenta $M = 128$ símbolos o clases, con probabilidad uniforme de $p_{ij} = 0.0078125$:

$$L = - \sum_{i=1}^{128} (y_{ij} \ln(0.0078125)) = 4.85 \quad (5.13)$$

El valor inicial de pérdida observado en la figura 5.81 es ≈ 2.5 al inicio del entrenamiento (casi la mitad del valor calculado), decrece y se estabiliza ≈ 1 ; por otro lado, los valores de pérdida para la validación comienza en 1.4 y decrece a ≈ 0.2 luego de 30 épocas. Para la gráfica de precisión, empieza con un valor ≈ 0.3 y alcanza ≈ 0.62 en el entrenamiento; sin embargo, para el set de validación comienza en ≈ 0.78 y se estabiliza en ≈ 1 ; este comportamiento se debe a la tasa empleada que es mayor a 1 en la cual se está eliminando información para cada símbolo; es decir, la red no es capaz de ajustar los pesos para permitir minimizar los errores. Esta tasa se considera únicamente para comparación y efectos de visualización para el diagrama de constelación.

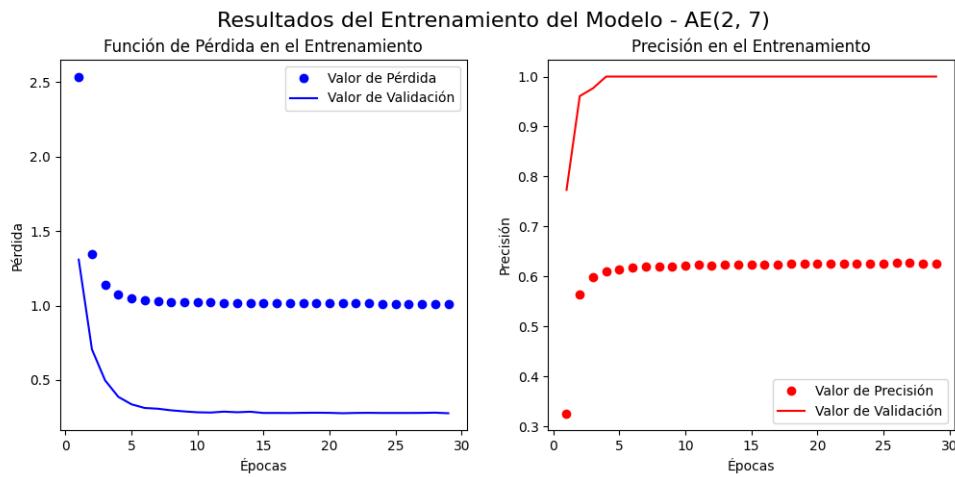


Figura 5.81: Resultados del entrenamiento para el auto codificador (2,7) normalizado en potencia.

El segundo caso de análisis es para el auto codificador (7,7), que produce una tasa de código $R = 1$, es decir, no se agrega redundancia. En la figura 5.82 el valor inicial de pérdida, en el entrenamiento, es ≈ 0.6 y decrece hasta 0.0 rápidamente hasta 17 épocas; mientras, el valor de pérdida para la validación empieza en ≈ 0 . En la gráfica de la precisión, en el entrenamiento, se observa que inicia con un valor de 0.95 y alcanza ≈ 1 . Para la validación en un inicio alcanza ≈ 1 , este comportamiento se debe a que a pesar de que no exista redundancia de bits el ajuste de los pesos ayuda a que el modelo realice de mejor manera la generalización y este listo para cualquier conjunto de datos, evitando el sobre ajuste.

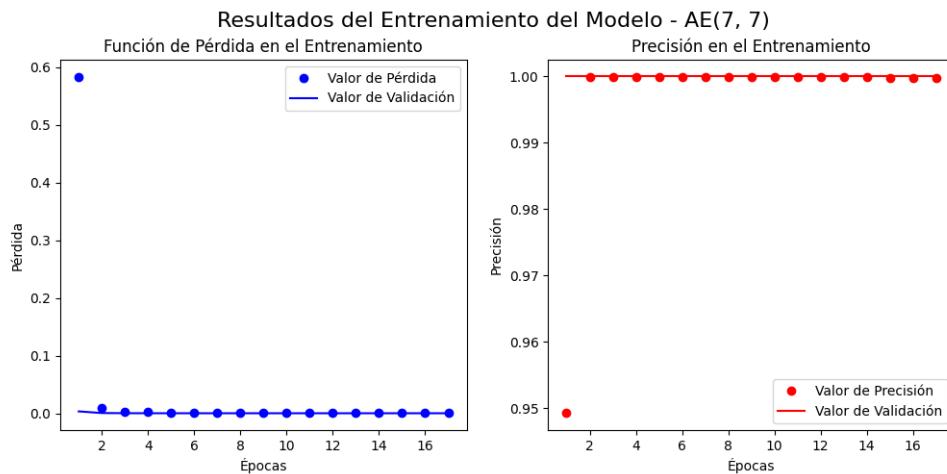


Figura 5.82: Resultados del entrenamiento para el auto codificador (7,7) normalizado en potencia.

El último caso, es para el auto codificador (14,7), que produce una tasa de código $R = \frac{1}{2}$,

es decir, se agrega redundancia del 50 %. En la figura 5.83 el valor inicial de pérdida, en el entrenamiento, es ≈ 0.4 y decrece hasta 0 en 17 épocas (pérdida reducida en un 20 % en comparación a la tasa $R = 1$ al iniciar el entrenamiento); mientras, que el valor de pérdida para la validación empieza relativamente en ≈ 0 . En la gráfica de la precisión, para los datos de entrenamiento, el valor inicial es 0.97 y alcanza ≈ 1 (≈ 2 % más que la tasa $R = 1$). Para la validación alcanza ≈ 1 rápidamente, esto se obtiene como resultado de emplear más canales adicionales que los casos anteriores.

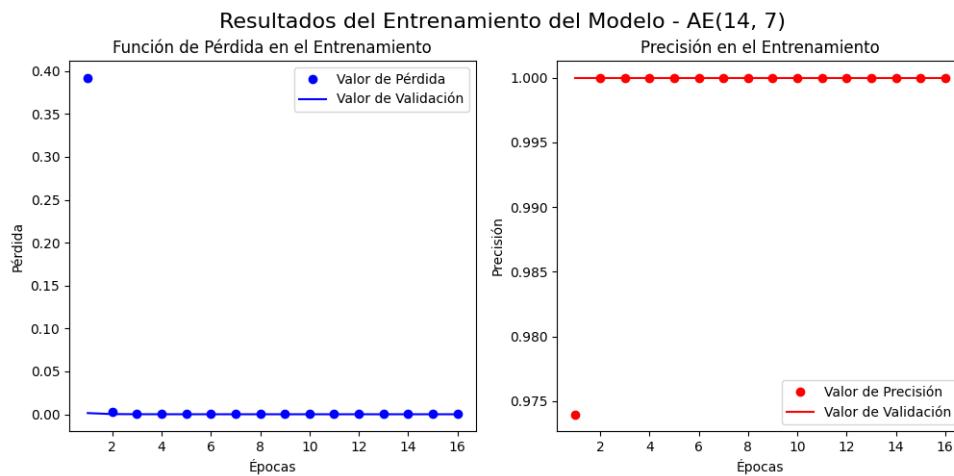


Figura 5.83: Resultados del entrenamiento para el auto codificador (14,7) normalizado en potencia.

En conclusión, los mejores entrenamientos que se obtuvieron son con las tasas $R = 1$ y $R = \frac{1}{2}$ ya que se logró llegar al mínimo en la pérdida y el máximo en la precisión, ambos parámetros tanto para datos de entrenamiento como validación. Siendo la excepción en la tasa $R = 3.5$ ya que al ser mayor a 1 la información no se envía completa. Sin embargo, como se ha mencionado anteriormente el aumento de canales genera la necesidad de un mayor tiempo y recursos para el entrenamiento tal como se muestra en la tabla 5.9.

5.11.2. Diagrama de Constelación

En la figura 5.84 se observa el diagrama de constelación para el auto codificador (2,7) el cual converge a una modulación 128-Quadrature Amplitude Modulation (128-QAM). La figura de la izquierda muestra una constelación sin la presencia de ruido (salida del codificador); para este caso, los 128 símbolos se encuentran exactamente en sus posiciones ideales según la red neuronal. Para un mayor detalle, en el apéndice B, tablas B.8 - B.9 se observan los valores de salida de la parte del codificador.

En las otras dos constelaciones se introduce ruido, es decir, es la representación de los símbolos que entrarán en el decodificador. En la figura del centro, el diagrama muestra la misma señal modulada de la izquierda, pero con un nivel de señal a ruido correspondiente a 20 dB, el ruido introduce perturbaciones en la posición de cada uno de los 128 símbolos, lo que causa que se produzca la interferencia entre los mismos tal como se muestra en la constelación; consiguiendo aumentar la probabilidad de error al momento de querer reconstruirlos.

Por último, la figura de la derecha muestra la constelación que corresponde a un SNR de 30 dB. El nivel de ruido es menor que en el caso de 20 dB; resultando en una menor dispersión de cada uno de los 128 símbolos alrededor de sus posiciones ideales, para este nivel de SNR se observa que la probabilidad de una interferencia entre los símbolos es casi nula para los que se encuentran en las afueras y una probabilidad considerable para los símbolos que se encuentran rodeando el origen. Se nota además que para este auto codificador, el valor de SNR utilizado en las constelaciones es mayor debido al incremento de símbolos y por ende a una mejor apreciación y distinción del efecto del ruido en estos.

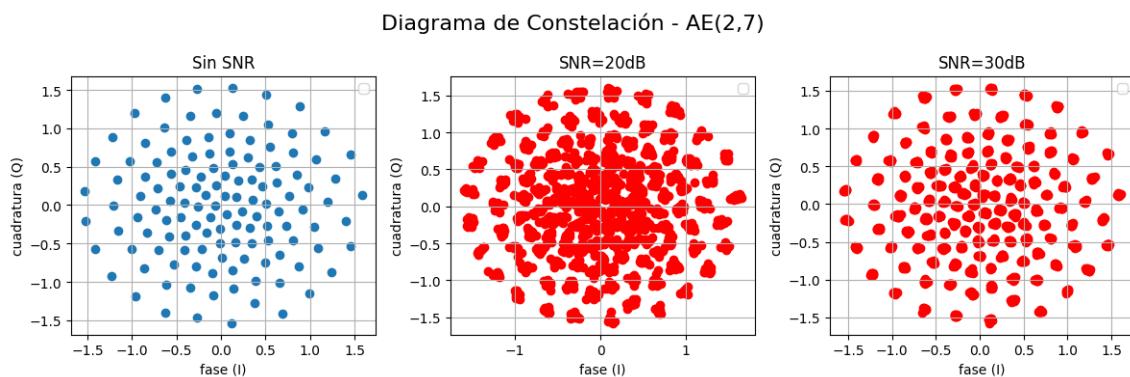


Figura 5.84: Diagrama de constelación para el auto codificador (2,7) normalizado en potencia.

A continuación se presentan las proyecciones correspondientes a las tasas $R = 1$ y $R = \frac{1}{2}$; las cuales tiene más de 2 canales a la salida del codificador.

En las figuras 5.85 y 5.86 se observa la reducción de dimensionalidad para 7 y 14 canales respectivamente, en las subfiguras de la izquierda se utilizó el algoritmo de t-SNE en python en el cual se observa que al contrario de las modulaciones anteriores este sigue una estructura parecida a la de la figura 5.75, teniendo en consideración que para tasa $R = 1$ la escala de valores se encuentra reducida a \approx la mitad. Mientras que para las subfiguras de la derecha se utilizó el algoritmo de UMAP en python, el cual ofrece un resultado de estructura igualmente parecida a la estructura modelo pero con un ligero grado de desventaja en cuanto a similitud con respecto a t-SNE especialmente para tasa $R = \frac{1}{2}$. Estas diferencias que se observan es

debido a como cada algoritmo toma los datos para representarlos en la dimensión 2, cabe recalcar que los valores mostrados en las diferentes proyecciones no representan los valores originales en los cuales se encuentras los símbolos para dimensiones mayores a 2.

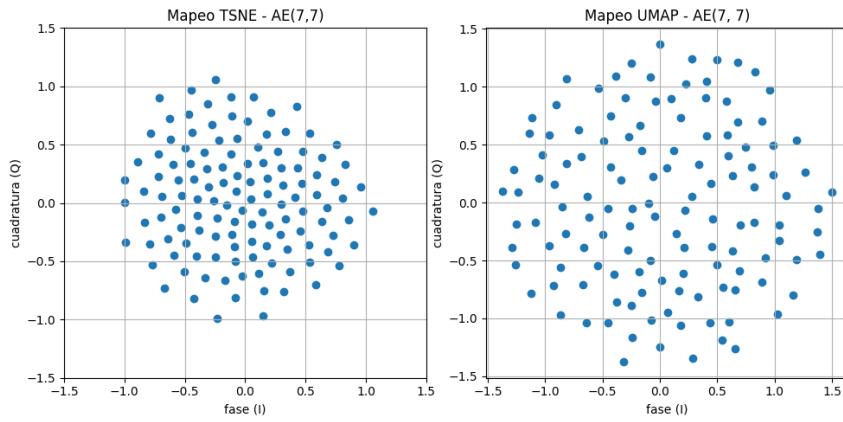


Figura 5.85: Constelaciones proyectadas para el auto codificador (7,7) normalizado en potencia.

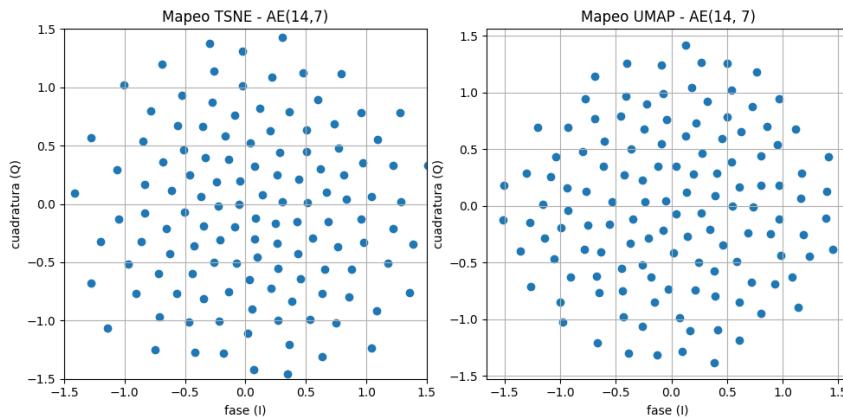


Figura 5.86: Constelaciones proyectadas para el auto codificador (14,7) normalizado en potencia.

5.11.3. Probabilidad de Error

En la figura 5.87 se muestra la gráfica del BLER y la Relación Señal-Ruido por bit (E_b/N_0) para el auto codificador (n,7) y la modulación 128-QAM. La gráfica izquierda corresponde a una escala lineal, donde se aprecia cómo el BLER disminuye a medida que aumenta el E_b/N_0 para las diferentes tasas de código. Como se mencionó previamente la tasa $R = 3.5$ (AE(2,7)) no es usada por lo que su BLER es el inferior de todos los casos. Al variar la tasa de código a $R = 1/2$ (AE(14,7)) se puede observar que tienen un mejor rendimiento que con tasa $R = 1$,

como es de esperarse por la redundancia. Para todas las curvas los IC tienen valores muy pequeños que están en el rango de 10^{-4} hasta 10^{-7} , los cuales se aprecian de manera más detallada en el apéndice C, tabla C.8. Estos valores mínimos nos indican que la dispersión de los datos es mínima.

En la gráfica derecha, que muestra la relación BLER vs. E_b/N_0 en escala logarítmica, se compara el rendimiento de un sistema con auto codificador (n,7) frente a un sistema 128-QAM sin codificación. Inicialmente, para valores muy bajos de E_b/N_0 , el AE con $R = \frac{1}{2}$ muestra un desempeño mejor que el sistema tradicional (128QAM(7,7)); sugiriendo la capacidad de corrección de errores en valores bajos de E_b/N_0 . La curva del auto codificador (7,7), demuestra un mejor rendimiento que el sistema base; esto se debe a que, aunque el sistema no tiene codificación de bloque, ha aprendido a decodificar de manera más eficiente la información, superando al sistema 128-QAM estándar. Mientras que para la tasa $R = 3.5$ (AE(2,7)), el rendimiento es mucho inferior que para la modulación 128-QAM convencional debido a que en esta ocasión no se está incorporando redundancia, si no al contrario quitando información (de los 7 bits solo se envían 2 bits).

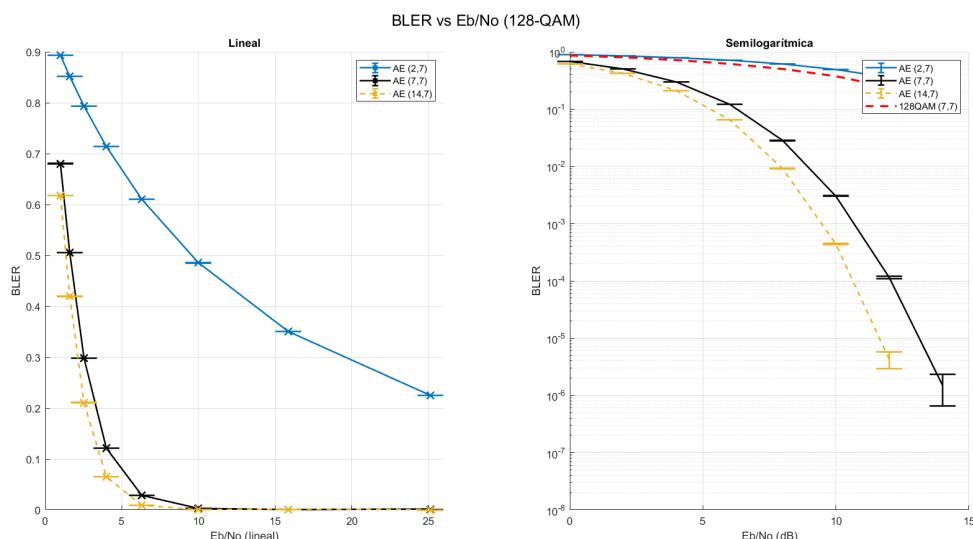


Figura 5.87: Gráfica del BLER para auto codificador (n,7) normalizado en potencia.

Aunque para 0dB el comportamiento es casi el mismo para el auto codificador con $R = 1$ y con $R = \frac{1}{2}$, a medida que va creciendo los valores de E_b/N_0 las tasas más pequeñas del AE tienen un mejor resultado, siendo el auto codificador (14,7) el que mejor rendimiento presenta llevando una ventaja de $\approx 1\text{dB}$ en comparación con el auto codificador (7,7); esto se debe a la redundancia que posee el primero (7 bits de redundancia).

Sin embargo, con tasas de codificación más bajas, se necesita transmitir más bits para enviar la misma cantidad de información útil, lo que puede reducir la eficiencia espectral. Utilizar más ancho de banda para transmitir la misma cantidad de datos puede ser problemático en canales con ancho de banda limitado. Este análisis se amplía en la sección [5.12](#).

A continuación para analizar el comportamiento del BER para este auto codificador ($n,7$). En el apéndice D, tabla D.8 se observan los valores de BER con sus respectivos intervalos de confianza, los cuales están en el rango de 10^{-4} hasta 10^{-7} , esto indica una dispersión mínima de los datos. En la figura 5.88 se muestran las curvas en escala lineal y logarítmica. Se nota que el auto codificador (2,7) presenta el inferior comportamiento (mayor bits erróneos) que todas las modulaciones incluyendo al sistema 128-QAM sin codificación, esto debido a la falta de información que presenta este (de 7 bits se envían solo 2). Se hace notar que el auto codificador (7,7) tiene una mayor probabilidad de error que el sistema 128-QAM sin codificación hasta $\approx 2dB$ (punto de intersección), en adelante el AE presenta una mejora; esto se debe a que, aunque el sistema no tiene codificación, ha aprendido a decodificar de manera más eficiente la información. Para las otras tasas de codificación se observa un mejor rendimiento, teniendo el mejor comportamiento la tasa de $R = \frac{1}{2}$ (redundancia de 7 bits), llevando una ventaja de $\approx 1dB$ a comparación con el AE de tasa $R = 1$ (sin redundancia). Al comparar estos resultados con la gráfica de BLER (figura 5.87) es visible que las curvas de BER empiezan en un valor más bajo indicando que la probabilidad de error a nivel de bit es menor. Si el BER es menor al BLER se puede llegar a la conclusión que hay pocos bits erróneos en general, esos errores están distribuidos de manera que afectan a muchos bloques y en consecuencia el BLER aumenta.

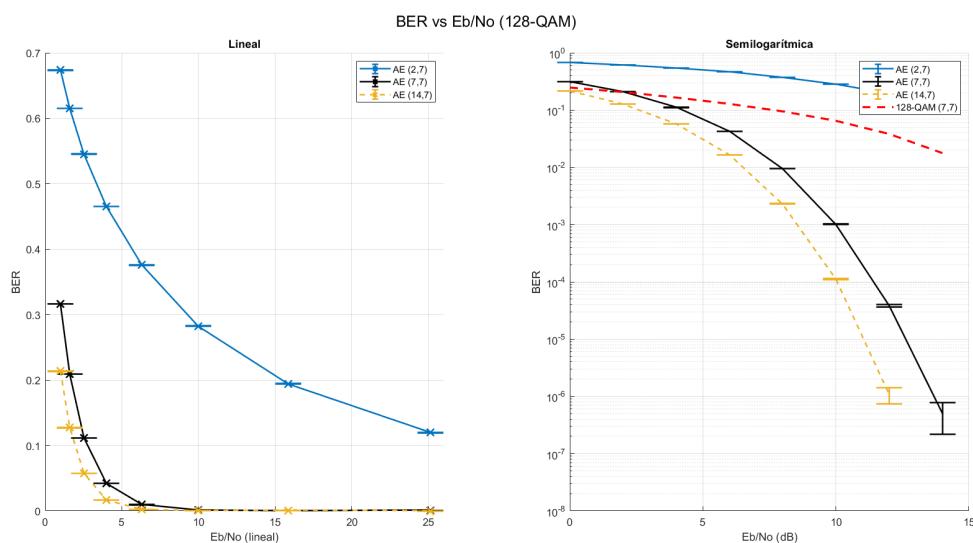


Figura 5.88: Gráfica del BER para auto codificador ($n,7$) normalizado en potencia.

5.12. Resumen de resultados y Discusión

5.12.1. Análisis de los Resultados de Entrenamiento

En la tablas 5.8 y 5.9 se presenta a manera de resumen los resultados de parámetros como tiempo, épocas, pérdida y validación en el entrenamiento y en la validación para las diferentes configuraciones de auto codificadores analizados y que fueron entrenados con la capa de normalización tanto de energía como de potencia.

Las gráficas de entrenamiento observadas para los distintos auto codificadores demuestran una clara tendencia a lograr resultados óptimos. El primer y más importante objetivo en el entrenamiento es minimizar la función de pérdida, esta función es la responsable de medir la diferencia entre la salida del auto codificador y la entrada original, proporcionando una métrica cuantitativa para evaluar qué tan bien el modelo está aprendiendo a reconstruir la señal original después de la compresión y la transmisión a través del canal ruidoso. Por lo tanto, este valor debe alcanzar el cero (0), o lo más cercano, mediante la actualización iterativa de los pesos en la red neuronal y un proceso de control para terminar el entrenamiento cuando no se ha podido reducir el valor luego de 5 épocas, evitando la oscilación entre valores pequeños de los pesos.

El segundo objetivo de maximizar la precisión de la red, permite decidir correctamente qué información se ha recibido a partir de los valores afectados por el ruido del canal; es decir, permite al auto codificador clasificar correctamente las señales recibidas y asignarlas a las categorías correctas de datos originales. Por lo tanto, la precisión en este contexto debe alcanzar el 100 %

Tabla 5.8: Resultados finales del entrenamiento para los diferentes auto codificadores normalizados en energía.

Modulación	n	k	R	Tiempo segundos	Épocas	Entrenamiento		Validación	
						Precisión máxima %	Pérdida mínima	Precisión máxima %	Pérdida mínima
BPSK	2	1	1/2	6.69	16	91.66	0.2086	100	0.0251
QPSK	2	2	1	31.92	23	84.85	0.3875	100	0.0368
	3	2	2/3	48.96	27	87.82	0.3282	100	0.0140
	4	2	1/2	36.76	27	87.80	0.3282	100	0.0150
	2	3	3/2	26.89	16	65.13	0.8203	100	0.2898
8-PSK	3	3	1	42.38	27	78.81	0.5735	100	0.0455
	4	3	3/4	37.02	26	83.34	0.4764	100	0.0148
	6	3	1/2	60.98	41	84.71	0.4382	100	0.0114
	2	4	2	25.08	15	41.74	1.3561	100	0.8235
16-PSK	4	4	1	25.26	15	74.15	0.7437	100	0.0437
	6	4	2/3	30.48	29	80.25	0.5899	100	0.0114
	8	4	1/2	68.46	40	83.44	0.5106	100	0.0047
	2	5	5/2	31.55	15	23.93	1.9241	81.25	1.4054
32-PSK	5	5	1	57.21	46	86.70	0.3980	100	0.0016
	6	5	5/6	58.63	22	89.59	0.3151	100	0.0005
	10	5	1/2	71.98	40	93.74	19.43	100	0.0001

Tabla 5.9: Resultados finales del entrenamiento para los diferentes auto codificadores normalizados en potencia.

Modulación	n	k	R	Tiempo segundos	Épocas	Entrenamiento		Validación	
						Precisión máxima %	Pérdida mínima	Precisión máxima %	Pérdida mínima
4-QAM	2	2	1	24.92 s	10	94.85	0.152	100	0.0005
	3	2	2/3	27.09 s	14	97.72	0.087	100	0.00012
	4	2	1/2	37.40 s	15	97.52	0.085	100	0.00005
16-QAM	2	4	1	19.50 s	13	54.40	1.2076	100	0.3976
	4	4	1	27.91 s	19	74.74	0.7460	100	0.0043
	6	4	2/3	26.45 s	16	82.29	0.5917	100	0.0017
	8	4	1/2	31.68 s	21	83.24	0.5129	100	0
32-QAM	2	5	5/2	30.44 s	18	54.83	1.335	100	0.5778
	5	5	1	35.22 s	26	86.54	0.4051	100	0.0005
	6	5	5/6	68.08 s	30	89.51	0.3183	100	0.00004
	10	5	1/2	48.63 s	18	93.69	0.1958	100	0
64-QAM	2	6	3	49.06 s	17	67.98	0.8888	100	0.2029
	6	6	1	58.39 s	21	99.96	0.031	100	0
	9	6	2/3	69.53 s	25	100	0	100	0
	12	6	1/2	131.77 s	18	100	0	100	0
128-QAM	2	7	7/2	330.11 s	30	62.56	1.012	100	0.2838
	7	7	1	164.58 s	18	100	0	100	0
	14	7	1/2	183.72 s	16	100	0	100	0

5.12.2. Análisis de los Resultados de los Diagramas de Constelación

Los diagramas de constelación en los auto codificadores ofrecen una representación visual de cómo los símbolos son mapeados en el espacio tras la optimización en el entrenamiento.

A diferencia de los esquemas de modulación digital tradicionales, que tienen diagramas de constelación predefinidos y fijos, las constelaciones de los auto codificadores fueron generados de manera dinámica en función de dos parámetros: la optimización de la función de pérdida y el nivel de ruido del canal. Este análisis se centrará en la interpretación y las implicaciones de estos diagramas de constelación obtenidos.

La optimización de la función de pérdida es responsable de ajustar los parámetros del auto codificador para minimizar la distancia entre los puntos transmitidos y recibidos. Este ajuste permite a la red neuronal determinar la mejor posición de los símbolos en el espacio, con el fin de minimizar el error. Como se observó en las gráficas, en algunos casos la distribución de los símbolos no era uniforme, lo cual en los sistemas convencionales representa un problema significativo durante la decodificación y toma de decisiones. Sin embargo, en el caso del AE, que aborda el problema como un clasificador multiclase, la red es capaz de decodificar la información recibida de manera eficiente, incluso cuando los símbolos están muy próximos entre sí. Esto se debe a que el AE aprende a diferenciar entre las clases de manera efectiva, permitiendo una decodificación precisa a pesar de la proximidad entre los símbolos.

Adicionalmente, el nivel de ruido durante el entrenamiento juega un papel importante en la optimización de la recuperación de la información. Con buenas condiciones de canal, el AE puede representar los símbolos de manera más precisa, incluso logrando una distancia uniforme entre ellos. Sin embargo, su rendimiento en la clasificación disminuye con niveles altos de ruido. Por lo tanto, a medida que el nivel de ruido aumenta, el AE debe adaptarse para manejar la mayor distorsión y aún así recuperar los datos originales con una mayor precisión, lo cual sin duda puede generar una distribución diferente de los símbolos en el espacio.

5.12.3. Análisis de los Resultados de BLER

La tasa de errores de bloque (BLER) es una métrica importante en la evaluación del desempeño de los sistemas de comunicación, tanto en los esquemas de modulación tradicionales como para los auto codificadores. En los esquemas de modulación digital tradicionales, como QPSK o 16-QAM, el BLER está influenciado por la relación señal a ruido (SNR), la eficiencia del esquema de modulación y la capacidad de corrección de errores. Estos sistemas utilizan diagramas de constelación fijos y predefinidos lo que puede resultar en un mayor BLER en condiciones adversas.

Como se mencionó anteriormente en los objetivos del entrenamiento, una alta precisión en la

red neuronal asegura que la clasificación sea correcta, incluso en presencia de ruido. Esta optimización en la recuperación de datos reduce la probabilidad de errores de bloque. Por lo tanto, los auto codificadores entrenados muestran un desempeño superior en términos de BLER y robustez frente al ruido del canal. Las curvas de BLER vs. E_b/N_0 presentadas en las gráficas indican que los auto codificadores pueden operar eficientemente a menores relaciones de señal a ruido gracias a la incorporación de redundancia que permite una mejor decodificación.

5.12.4. Análisis de los Resultados de BER

En el contexto de uso de auto codificadores, también es fundamental analizar el BER porque permite evaluar detalladamente el rendimiento del auto codificador en el manejo de errores a nivel de bits. El BER proporciona una visión más granular y precisa de cómo el sistema gestiona las perturbaciones y errores individuales que ocurren durante la transmisión. Esta información es de gran relevancia para optimizar el diseño del AE y mejorar su capacidad para recuperar la información transmitida con alta fidelidad, especialmente en condiciones de canal adversas con niveles de ruido elevados.

Al analizar las diferentes gráficas de BER, se observa que el uso de auto codificadores generalmente mejora el rendimiento, reduciendo la probabilidad de error en la mayoría de los casos. En particular, a valores bajos de SNR, se evidencia una menor probabilidad de error en comparación con los esquemas tradicionales. Esto se debe a la capacidad de la red para detectar y corregir errores, así como para reconstruir la información original. Sin embargo, el ruido sigue siendo un desafío que impide alcanzar rendimientos aún mayores. Cabe destacar que no todo son ventajas, ya que, como se analizará en la siguiente sección, el uso de tasas mas pequeñas implica un costo en términos de ancho de banda y procesamiento.

5.12.5. Análisis sobre requerimientos de Ancho de Banda

El ancho de banda de una señal M-ary Phase Shift Keying (M-PSK) puede aproximarse por la siguiente ecuación:

$$B_{M-PSK} = \frac{2}{T} = \frac{2R_b}{\log_2(M)} \quad (5.14)$$

Donde:

- T : tiempo del símbolo.

- R_b : es el bit rate.
- M : orden de modulación.

La eficiencia del ancho de banda para M-PSK se puede expresar como:

$$\rho = \frac{R_b}{B} = \frac{\log_2(M)}{2} \quad (5.15)$$

Tabla 5.10: Ancho de banda eficiente para M-PSK.

M	2	4	8	16	32
ρ (bps/Hz)	0.5	1	1.5	2	2.5

En base a la Ecuación 5.15, en la tabla 5.10 se observan los valores de ancho de banda eficiente para M-PSK. A medida que M aumenta, la eficiencia espectral (bits por Hz) aumenta, de 0.5 hasta 2.5 bps/Hz, porque se transmiten más bits por símbolo. Sin embargo, el requisito de relación señal a ruido (SNR) también aumenta con el objetivo de mantener la misma tasa de error de bit (BER).

En el caso de M-ary Quadrature Amplitude Modulation (M-QAM) el ancho de banda se define como:

$$B_{M-QAM} = \frac{R_b}{\log_2(M)} \quad (5.16)$$

De esta forma la eficiencia espectral para M-QAM se expresa como:

$$\rho = \frac{R_b}{B} = \log_2(M) \quad (5.17)$$

La tabla 5.11 contiene valores de ancho de banda eficiente para M-QAM, si M aumenta también se incrementa la eficiencia espectral, de 2 hasta 7 bps/Hz, aún más que en el caso de M-PSK.

Tabla 5.11: Ancho de banda eficiente para M-QAM.

M	4	16	32	64	128
ρ (bps/Hz)	2	4	5	6	7

Para el caso del auto codificador, la estrategia para proteger los datos contra posibles errores que puede introducir el canal, implica introducir redundancia en los mensajes que se transmi-

ten. Al usar codificación de canal, la tasa efectiva aumenta, ya que se están transmitiendo más bits (datos + redundancia). En consecuencia, se produce un aumento del ancho de banda ya que está directamente relacionado con la tasa de bits. En este contexto, se transmiten k bits, sin codificación, en un tiempo T , ahora el AE debe transmitir n bits codificados; es decir, la duración de un bit pasa de ser $\frac{T}{k}$ a $\frac{T}{n}$ y el ancho de banda aumenta aproximadamente en un factor $\frac{n}{k}$. En conclusión, el aumento del ancho de banda coincide con el inverso de la tasa del código, a este factor se le conoce como factor de expansión (f_B).

$$f_B = \frac{\text{ancho banda codificado}}{\text{ancho banda no codificado}} = \frac{1}{R} = \frac{n}{k} \quad (5.18)$$

Donde:

- R : es la tasa de código.
- n : número de bits más la redundancia.
- k : número de bits en cada símbolo.

En la tabla 5.12 se observa el factor de expansión requerido para las diferentes tasas de código que se emplearon. En el caso de tasas que no tiene codificación, $R = 1$, el $f_B = 1$ indica que el ancho de banda no se incrementa; mientras que al reducir la tasa se puede requerir hasta un $f_B = 2$ que implica el doble de ancho de banda. En general, proteger la información implica usar códigos con tasas menores a 1 lo que implica un mayor coste en ancho de banda debido a la redundancia añadida.

Tabla 5.12: Factor de expansión en el Ancho de Banda al emplear codificación de canal.

R	1	$\frac{5}{6}$	$\frac{3}{4}$	$\frac{2}{3}$	$\frac{1}{2}$
factor expansión	1	1.2	1.33	1.5	2

Finalmente, se llega a la conclusión de que la variación del ancho de banda para una modulación M-PSK o M-QAM se vería afectado por el inverso de la tasa de código utilizada, es decir, si se utiliza una tasa de $R = \frac{1}{2}$, el ancho de banda requerido llegaría a ser el doble del habitual.

$$B_{M-PSK}(\text{codificado}) = \frac{2R_b}{\log_2(M)} \times \frac{n}{k} = \frac{2nR_b}{k^2} \quad (5.19)$$

$$B_{M-QAM}(\text{codificado}) = \frac{R_b}{\log_2(M)} \times \frac{n}{k} = \frac{nR_b}{k^2} \quad (5.20)$$

6. Conclusiones y recomendaciones

En este Capítulo se detallan las conclusiones a las que se ha llegado al finalizar este trabajo de titulación, partiendo de los objetivos planteados. Además, se realizan algunas recomendaciones en base a la experiencia obtenida durante el desarrollo del mismo. Finalmente, se sugieren temas de análisis para investigaciones e implementaciones futuras.

6.1. Conclusiones

A partir de la investigación previa se ha establecido los criterios de diseño para un auto codificador Autoencoder (AE) capaz de procesar la codificación y modulación de hasta 32 símbolos para energía constante (M-PSK) y 128 símbolos para energía variable (M-QAM), con el objetivo de minimizar los efectos que introduce el canal de comunicaciones.

La selección de funciones de activación como Rectified Linear Unit (ReLU), lineal y softmax influye en la capacidad y rendimiento de la red neuronal tanto para la codificación como para la decodificación. ReLU introduce no linealidad y mitiga el problema del desvanecimiento del gradiente, facilitando el aprendizaje de representaciones complejas. La función lineal permite salidas continuas, que pueden ser normalizadas en energía o potencia, al tiempo que previene la saturación de activaciones y mejora la estabilidad durante la convergencia. Finalmente, La función softmax proporciona distribuciones de probabilidad que se emplea para la decodificación de cada representación del símbolo o clase.

Durante el entrenamiento del AE, el objetivo principal es reducir la función de pérdida para mejorar la capacidad del modelo en la recuperación precisa de la información original. Para lograr este objetivo, se implementa la técnica de parada temprana (*Early stopping*), que previene el sobreajuste y guarda los mejores pesos del modelo entrenado. Estos pesos óptimos se utilizan durante la etapa de predicción para obtener los mejores resultados; es decir, este proceso asegura que los datos transmitidos se puedan reconstruir con precisión en el receptor. A medida que la red neuronal se entrena, no solo aprende a minimizar la función de pérdida, sino también a identificar patrones de error introducidos por el canal de comunicación. Este aprendizaje permite que el AE no solo optimice la transmisión de datos, sino que también implemente mecanismos de corrección de errores, mejorando así la confianza de la señal recibida.

La detección de errores se realiza mediante la comparación de los datos recibidos con las predicciones del auto codificador entrenado, permitiendo identificar diferencias que indican la

presencia de errores. Una vez detectados los errores, el auto codificador realiza correcciones para ajustar la señal recibida a la original; este proceso ayuda a la reducción en la tasa de errores de bloques (BLER) mejorando la calidad general de la comunicación. Esta mejora se evidencia especialmente en términos de Bit Error Rate (BER), dado que el AE es capaz de generar una probabilidad de error menor que la de los sistemas tradicionales, especialmente en tasas bajas de Signal to noise ratio (SNR) donde la señal transmitida está más afectada por el ruido, lo que aumenta la probabilidad de que se produzcan errores en la transmisión. Estas mejoras sin duda contribuyen a reducir el número de retransmisiones que tenga que realizar el sistema.

Los resultados de los auto codificadores muestran una tendencia prometedora hacia la optimización en comparación con los esquemas de modulación digital tradicionales, pero a pesar de sus beneficios la implementación de auto codificadores en sistemas de comunicación puede enfrentar ciertas limitaciones y desafíos. Uno de los principales es el mayor coste computacional requerido, asociado al entrenamiento e inferencia al usar tasas de codificación cada vez más bajas. En este contexto, si bien la redundancia de información al usar tasas más bajas conlleva una menor probabilidad de error, se requiere mayor ancho de banda.

El AE también puede necesitar grandes cantidades de memoria para almacenar los parámetros del modelo y los datos de entrenamiento; esto podría ser un problema en dispositivos con recursos limitados de memoria, como dispositivos IoT o dispositivos móviles. Además, el proceso de entrenamiento de redes neuronales profundas puede ser intenso en términos de energía, lo que puede afectar a la duración de la batería en dispositivos o la eficiencia energética de los sistemas integrados.

6.2. Recomendaciones

Para el procesamiento de los datos es necesario disponer de equipos con suficiente capacidad de procesamiento, preferiblemente equipada con Graphics Processing Unit (GPU), para acelerar el entrenamiento y la inferencia de los auto codificadores. Se recomienda el uso de la GPU para las operaciones intensivas en términos de cálculo, como el entrenamiento de la red neuronal, y la Central Processing Unit (CPU) para tareas de preprocesamiento y gestión de datos.

Además, se requiere una memoria RAM de alta capacidad para manejar grandes conjuntos de datos los cuales ayudan a una mejor precisión en los resultados del modelo implementado. En

este trabajo se empleó como técnica de gestión de memoria eficiente el uso de *mini-batches* durante el entrenamiento, para evitar sobrecargar la memoria disponible y asegurar un rendimiento óptimo.

6.3. Trabajos Futuros

En esta trabajo de titulación se ha demostrado el beneficio de usar un auto codificador, por esta razón se ha considerado que un área prometedora para investigaciones futuras se centraría en el desarrollo e implementación de estos modelos en plataformas de hardware. El objetivo sería determinar los dispositivos que ofrecen un rendimiento óptimo, así como identificar posibles limitaciones o incompatibilidades que puedan surgir durante la implementación. Esta investigación proporcionaría indicaciones específicas acerca de los desafíos asociados con la ejecución de AE en plataformas de hardware, lo que sería un factor importante para su adopción en sistemas de comunicación reales.

Para abordar las limitaciones de memoria y energía, se requiere investigar técnicas de compresión de modelos, optimización de memoria y métodos de entrenamiento energéticamente eficientes. Esto se considera un punto importante de análisis de la viabilidad de emplear auto codificadores y su implementación exitosa en sistemas de comunicación con restricciones de recursos.

El modelo implementado aborda la optimización únicamente entre un transmisor y receptor; sin embargo, sería necesario investigar y analizar técnicas que mejoren la capacidad del AE para resistir interferencias externas, como las señales generadas por dispositivos cercanos. Mejorar la resistencia a interferencias externas es fundamental para garantizar la robustez de los auto codificadores en entornos de comunicación reales y con múltiples dispositivos operando simultáneamente.

Finalmente, se puede abordar temas de seguridad y privacidad relacionados con la transmisión y almacenamiento de datos sensibles basado en AE, incluyendo la exploración de técnicas de cifrado y privacidad.

Apéndice A

A.1. Ejemplo de codificación *One Hot*

Tabla A.1: Ejemplo de codificación *One Hot* para 16-QAM

ID	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Apéndice B

B.1. Valores para el diagrama de constelación de auto codificadores (2,k) normalizados en energía.

Tabla B.1: Valores del auto codificador (2,3) normalizado en energía.

S1	-0.5162	-0.8564	S5	-0.2456	0.9693
S2	0.9665	0.2563	S6	-0.9719	-0.2352
S3	0.2406	-0.9706	S7	-0.8613	0.5079
S4	0.8614	-0.5078	S8	0.5244	0.8514

Tabla B.2: Valores del auto codificador (2,4) normalizado en energía.

S1	0.82	0.06	S5	0.58	-0.81	S9	-0.17	-0.99	S13	0.53	0.85
S2	-0.54	-0.84	S6	-0.22	1.00	S10	-0.57	0.82	S14	0.24	-0.97
S3	0.98	0.22	S7	-0.98	-0.22	S11	0.17	0.98	S15	-0.82	-0.57
S4	-0.98	0.18	S8	0.99	-0.16	S12	-0.84	0.54	S16	0.84	-0.54

Tabla B.3: Valores del auto codificador (2,5) normalizado en energía.

S1	0.91	0.41	S9	0.10	-1.00	S17	0.90	0.21	S25	-1.00	0.09
S2	-0.15	-0.99	S10	-0.29	-0.96	S18	0.93	-0.36	S26	0.73	-0.68
S3	-0.94	0.34	S11	-0.75	0.66	S19	-0.43	0.90	S27	0.85	-0.53
S4	-0.88	0.48	S12	-0.65	-0.76	S20	-1.00	-0.05	S28	-0.48	-0.88
S5	-0.96	-0.27	S13	0.16	0.99	S21	0.99	-0.16	S29	0.68	0.73
S6	0.43	-0.90	S14	-0.62	0.79	S22	-0.90	-0.44	S30	-0.79	-0.61
S7	0.20	-0.98	S15	1.00	0.03	S23	0.35	0.93	S31	-0.03	1.00
S8	0.53	0.85	S16	0.81	0.59	S24	-0.24	0.97	S32	0.58	-0.82

B.2. Valores para el diagrama de constelación de auto codificadores (2,k) normalizados en potencia.

Tabla B.4: Valores del auto codificador (2,2) normalizado en potencia.

S1	-0.94	-0.34	S3	0.98	0.25
S2	0.31	-1.00	S4	-0.28	0.91

Tabla B.5: Valores del auto codificador (2,4) normalizado en potencia.

S1	-1.16	0.58	S5	-1.21	-0.26	S9	0.30	1.23	S13	0.82	-0.92
S2	0.03	0.63	S6	1.03	0.76	S10	0.01	0.03	S14	0.05	-1.22
S3	0.56	0.27	S7	0.49	-0.31	S11	-0.73	-0.96	S15	1.23	-0.12
S4	-0.51	-0.24	S8	-0.53	1.12	S12	-0.50	0.35	S16	-0.08	-0.57

Tabla B.6: Valores del auto codificador (2,5) normalizado en potencia.

S1	-0.31	0.79	S9	0.79	-0.24	S17	-0.32	-0.69	S25	1.13	0.67
S2	-0.12	0.40	S10	0.40	-0.10	S18	-0.73	-1.03	S26	0.68	1.11
S3	0.50	-0.59	S11	0.77	0.19	S19	-1.26	-0.16	S27	1.30	0.08
S4	0.27	0.29	S12	-0.80	0.01	S20	0.16	0.77	S28	-0.74	-0.42
S5	-1.21	0.41	S13	0.58	0.60	S21	0.09	-0.40	S29	-0.93	0.90
S6	1.24	-0.52	S14	-0.63	0.50	S22	0.89	-0.97	S30	-0.46	1.27
S7	-0.22	-1.28	S15	0.00	0.01	S23	-0.33	-0.28	S31	0.11	1.27
S8	0.38	-1.26	S16	0.10	-0.81	S24	-0.38	0.12	S32	-1.18	-0.70

Tabla B.7: Valores del auto codificador (2,6) normalizado en potencia.

S1	0.16	1.43	S17	1.50	0.08	S33	-0.46	-0.08	S49	-0.17	0.03
S2	0.38	-0.01	S18	-0.45	0.52	S34	-0.85	-0.63	S50	0.32	0.28
S3	-1.15	0.88	S19	-0.61	-0.88	S35	-0.47	0.22	S51	0.25	-0.66
S4	-0.56	-1.32	S20	0.50	-0.52	S36	1.27	-0.78	S52	-0.73	-0.08
S5	-0.21	0.21	S21	-1.03	-0.27	S37	-0.79	1.22	S53	-0.31	1.38
S6	0.56	0.86	S22	-0.67	0.75	S38	-1.38	0.40	S54	0.83	-0.25
S7	0.32	0.62	S23	-0.06	0.92	S39	0.53	0.40	S55	-0.05	0.38
S8	1.25	-0.33	S24	0.07	0.61	S40	0.61	0.08	S56	0.22	1.00
S9	0.55	-0.88	S25	-0.24	-0.25	S41	-0.62	-0.40	S57	0.09	0.20
S10	0.83	-0.61	S26	-1.06	0.12	S42	0.83	0.63	S58	0.13	-0.03
S11	-0.04	-0.66	S27	0.65	1.32	S43	1.07	1.01	S59	0.84	0.29
S12	-0.33	-0.69	S28	0.18	-1.04	S44	0.53	-0.21	S60	0.01	-0.43
S13	0.47	-1.37	S29	-0.08	-1.45	S45	-1.31	-0.61	S61	-1.01	-1.03
S14	-0.22	-1.02	S30	-0.34	-0.44	S46	-0.02	-0.17	S62	0.90	-1.11
S15	-0.73	0.26	S31	-0.92	0.49	S47	0.26	-0.33	S63	-1.45	-0.11
S16	1.28	0.52	S32	-0.24	0.61	S48	-0.39	0.93	S64	1.02	0.06

Tabla B.8: Valores del auto codificador (2,7) normalizado en potencia - Parte 1.

S1	0.36	0.07	S33	-0.35	1.16	S65	0.19	0.32	S97	0.53	-0.46
S2	-0.40	0.46	S34	0.98	0.23	S66	-0.94	-0.15	S98	0.35	0.33
S3	-1.01	-0.57	S35	0.25	1.16	S67	-0.21	-0.57	S99	0.23	-0.08
S4	-1.52	-0.21	S36	0.91	-0.06	S68	0.87	-0.80	S100	0.51	1.44
S5	-0.28	0.23	S37	0.50	0.02	S69	-1.53	0.18	S101	-0.58	-0.40
S6	-1.16	0.33	S38	-0.39	0.85	S70	-0.71	-0.58	S102	-0.13	0.68
S7	-0.28	-0.25	S39	0.30	0.62	S71	-0.14	0.88	S103	0.00	-0.68
S8	-0.63	-1.40	S40	-0.91	0.11	S72	0.62	0.51	S104	0.88	1.29
S9	0.13	-1.09	S41	-0.01	-0.31	S73	-0.73	-0.05	S105	0.85	0.40
S10	0.43	0.50	S42	-1.02	0.57	S74	0.76	0.13	S106	0.17	-0.26
S11	1.07	0.60	S43	0.12	-1.53	S75	-1.23	-0.92	S107	-1.22	0.89
S12	0.99	-1.15	S44	0.69	-1.42	S76	0.06	-0.12	S108	-0.06	1.19
S13	0.81	-0.45	S45	-0.97	1.20	S77	-1.16	-0.33	S109	-0.83	-0.36
S14	0.04	0.12	S46	0.13	-0.85	S78	0.53	1.05	S110	-0.26	-0.02
S15	0.26	-0.69	S47	-0.07	-0.00	S79	0.35	-0.50	S111	-0.18	0.13
S16	-0.05	0.25	S48	-0.96	-1.18	S80	-0.23	0.37	S112	0.13	1.53

Tabla B.9: Valores del auto codificador (2,7) normalizado en potencia - Parte 2.

S17	1.46	0.66	S49	0.51	-0.27	S81	-0.09	-1.18	S113	-0.40	-0.39
S18	0.16	-0.49	S50	-0.86	-0.83	S82	-0.61	-0.20	S114	-0.16	-0.36
S19	1.09	-0.56	S51	-0.56	0.69	S83	0.55	0.76	S115	0.37	-1.28
S20	-0.62	-1.03	S52	-0.85	0.37	S84	-0.08	0.49	S116	1.40	-0.21
S21	0.73	-0.27	S53	0.65	-1.01	S85	0.13	0.53	S117	0.47	0.24
S22	-0.85	0.81	S54	-0.53	-0.77	S86	0.49	-0.75	S118	0.39	-0.98
S23	0.66	-0.65	S55	-0.64	1.02	S87	-0.57	0.05	S119	-0.41	0.03
S24	1.16	0.96	S56	1.19	0.04	S88	-1.05	0.63	S120	0.27	0.64
S25	1.55	1.06	S57	-0.36	0.64	S89	0.31	-0.31	S121	0.17	0.11
S26	-0.47	0.25	S58	-0.27	-1.47	S90	0.62	0.29	S122	-1.21	-0.00
S27	1.46	-0.54	S59	1.58	0.13	S91	-0.35	-1.07	S123	-0.27	1.51
S28	0.40	-0.14	S60	-0.46	-0.15	S92	0.81	0.69	S124	-1.41	-0.58
S29	0.09	0.70	S61	-0.62	1.40	S93	0.10	0.94	S125	-0.10	-0.89
S30	0.05	0.38	S62	-0.56	0.41	S94	0.79	0.94	S126	-0.28	-0.80
S31	0.65	-0.08	S63	0.32	-0.30	S95	-0.72	0.56	S127	-0.71	0.22
S32	-0.32	0.64	S64	-0.41	-0.59	S96	1.24	-0.87	S128	-0.02	-0.50

Apéndice C

C.1. Valores de BLER para auto codificadores (n,k) normalizados en energía.

Tabla C.1: Valores de BLER con su respectivo IC para el auto codificador (n,3) normalizado en energía.

	(2,3)		(3,3)		(4,3)		(6,3)	
E_b/N_0 dB	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %
0	0,5022	$\pm 1,94E-04$	0,3991	$\pm 3,423E-04$	0,3593	$\pm 2,57E-04$	0,3438	$\pm 3,26E-04$
2	0,4033	$\pm 4,18E-04$	0,2740	$\pm 2,147E-04$	0,2287	$\pm 1,85E-04$	0,2132	$\pm 2,23E-04$
4	0,2940	$\pm 2,50E-04$	0,1531	$\pm 3,282E-04$	0,1118	$\pm 1,41E-04$	0,0997	$\pm 1,12E-04$
6	0,1872	$\pm 1,80E-04$	0,0618	$\pm 7,317E-05$	0,0355	$\pm 1,02E-04$	0,0299	$\pm 1,23E-04$
8	0,0972	$\pm 1,45E-04$	1,53E-02	$\pm 6,509E-05$	5,71E-03	$\pm 5,77E-05$	0,0044	$\pm 2,94E-05$
10	3,73E-02	$\pm 1,04E-04$	1,74E-03	$\pm 2,897E-05$	3,12E-04	$\pm 1,10E-05$	2,39E-04	$\pm 1,05E-05$
12	9,00E-03	$\pm 5,99E-05$	6,46E-05	$\pm 8,008E-06$	3,30E-06	$\pm 1,01E-06$	2,80E-06	$\pm 5,70E-07$
14	1,07E-03	$\pm 2,27E-05$	5,00E-07	$\pm 3,267E-07$	0	0	0	0

Tabla C.2: Valores de BLER con su respectivo IC para el auto codificador (n,4) normalizado en energía.

	(2,4)		(4,4)		(6,4)		(8,4)	
E_b/N_0 dB	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %
0	0,6958	$\pm 2,39E-04$	0,4845	$\pm 3,654E-04$	0,4317	$\pm 2,67E-04$	0,4034	$\pm 2,75E-04$
2	0,6243	$\pm 2,16E-04$	0,3354	$\pm 3,277E-04$	0,2741	$\pm 2,87E-04$	0,2422	$\pm 3,52E-04$
4	0,5385	$\pm 3,00E-04$	0,1852	$\pm 2,786E-04$	0,1295	$\pm 1,93E-04$	0,1030	$\pm 2,50E-04$
6	0,4409	$\pm 3,90E-04$	0,0712	$\pm 1,306E-04$	0,0377	$\pm 1,25E-04$	0,0249	$\pm 9,52E-05$
8	0,3343	$\pm 3,42E-04$	1,56E-02	$\pm 5,162E-05$	5,22E-03	$\pm 4,45E-05$	0,0024	$\pm 1,59E-05$
10	2,28E-01	$\pm 2,30E-04$	1,47E-03	$\pm 2,224E-05$	2,27E-04	$\pm 8,01E-06$	5,80E-05	$\pm 3,97E-06$
12	1,34E-01	$\pm 2,30E-04$	4,17E-05	$\pm 3,921E-06$	1,10E-06	$\pm 8,49E-07$	2,00E-07	$\pm 2,61E-07$
14	6,38E-02	$\pm 1,80E-04$	2,00E-07	$\pm 2,613E-07$	0	0	0	0

Tabla C.3: Valores de Block Error Rate (BLER) con su respectivo IC para el auto codificador (n,5) normalizado en energía.

		(2,5)		(5,5)		(6,5)		(10,5)	
E_b/N_0 dB	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %	
0	0,8280	$\pm 3,10E-04$	0,5510	$\pm 3,258E-04$	0,5223	$\pm 2,83E-04$	0,4671	$\pm 1,99E-04$	
2	0,7861	$\pm 2,11E-04$	0,3839	$\pm 4,503E-04$	0,3483	$\pm 3,00E-04$	0,2832	$\pm 3,55E-04$	
4	0,7352	$\pm 2,31E-04$	0,2069	$\pm 2,092E-04$	0,1742	$\pm 2,03E-04$	0,1188	$\pm 2,39E-04$	
6	0,6748	$\pm 2,76E-04$	0,0736	$\pm 1,544E-04$	0,0545	$\pm 1,35E-04$	0,0270	$\pm 1,15E-04$	
8	0,6059	$\pm 2,75E-04$	1,35E-02	$\pm 5,265E-05$	8,03E-03	$\pm 3,37E-05$	0,0023	$\pm 2,85E-05$	
10	5,31E-01	$\pm 4,12E-04$	8,88E-04	$\pm 1,583E-05$	4,13E-04	$\pm 5,37E-06$	4,71E-05	$\pm 4,24E-06$	
12	4,54E-01	$\pm 3,40E-04$	1,04E-05	$\pm 1,875E-06$	4,00E-06	$\pm 9,69E-07$	2,00E-07	$\pm 2,61E-07$	
14	3,84E-01	$\pm 3,00E-04$	1,00E-07	$\pm 1,960E-07$	0	0	0	0	

C.2. Valores de BLER para auto codificadores (n,k) normalizados en potencia.

Tabla C.4: Valores de BLER con su respectivo IC para el auto codificador (n,2) normalizado en potencia.

E_b/N_0 dB	(2,2)		(3,2)		(4,2)	
	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %
0	0,1519	$\pm 8,12E-05$	0,1223	$\pm 2,66E-04$	0,1209472	$\pm 1,85E-04$
2	0,0745	$\pm 2,57E-04$	0,0511	$\pm 1,76E-04$	0,0508244	$\pm 9,65E-05$
4	0,0256	$\pm 9,93E-05$	0,0133	$\pm 7,19E-05$	0,0133851	$\pm 7,70E-05$
6	0,0050	$\pm 2,70E-05$	0,0016	$\pm 2,42E-05$	1,69E-03	$\pm 2,64E-05$
8	4,47E-04	$\pm 1,42E-05$	6,39E-05	$\pm 3,87E-06$	7,22E-05	$\pm 6,63E-06$
10	1,13E-05	$\pm 2,00E-06$	4,00E-07	$\pm 3,20E-07$	5,00E-07	$\pm 3,27E-07$
12	1,00E-07	$\pm 1,96E-07$	0	± 0	0	± 0
14	0	± 0	0	± 0	0	± 0

Tabla C.5: Valores de BLER con su respectivo IC para el auto codificador (n,4) normalizado en potencia.

E_b/N_o dB	2,4		4,4		6,4		8,4	
	BLER media	IC 95 %						
0	0,6020	$\pm 1,50E-04$	0,4829	$\pm 2,77E-04$	0,4350	$\pm 3,46E-04$	0,4173	$\pm 3,37E-04$
2	0,5029	$\pm 2,72E-04$	0,3339	$\pm 3,19E-04$	0,2777	$\pm 3,77E-04$	0,2571	$\pm 2,23E-04$
4	0,3912	$\pm 3,79E-04$	0,1835	$\pm 2,61E-04$	0,1322	$\pm 1,37E-04$	0,1142	$\pm 2,36E-04$
6	0,2798	$\pm 2,33E-04$	0,0696	$\pm 1,40E-04$	0,0392	$\pm 1,10E-04$	0,0298	$\pm 9,00E-05$
8	0,1831	$\pm 2,47E-04$	0,0148	$\pm 5,27E-05$	0,0056	$\pm 2,20E-05$	0,0034	$\pm 2,39E-05$
10	0,1107	$\pm 1,43E-04$	0,0013	$\pm 2,13E-05$	0,0002	$\pm 5,69E-06$	0,0001	$\pm 6,77E-06$
12	0,0610	$\pm 8,49E-05$	2,81E-05	$\pm 2,16E-06$	2,00E-06	$\pm 6,53E-07$	4,00E-07	$\pm 3,20E-07$
14	0,0297	$\pm 1,30E-04$	1,00E-07	$\pm 1,96E-07$	0	0	0	0

Tabla C.6: Valores de BLER con su respectivo Intervalo de Confianza (IC) para el auto codificador (n,5) normalizado en potencia.

E_b/N_o dB	2,5		5,5		6,5		10,5	
	BLER media	IC 95 %						
0	0,7246	$\pm 2,54E-04$	0,5486	$\pm 2,57E-04$	0,5187	$\pm 2,36E-04$	0,4654	$\pm 2,97E-04$
2	0,6415	$\pm 2,88E-04$	0,3803	$\pm 2,81E-04$	0,3442	$\pm 2,14E-04$	0,2812	$\pm 1,98E-04$
4	0,5391	$\pm 2,59E-04$	0,2043	$\pm 2,74E-04$	0,1706	$\pm 1,98E-04$	0,1176	$\pm 1,56E-04$
6	0,4229	$\pm 3,78E-04$	0,0721	$\pm 2,17E-04$	0,0523	$\pm 1,33E-04$	0,0266	$\pm 8,44E-05$
8	0,3049	$\pm 3,82E-04$	0,0130	$\pm 3,39E-05$	0,0076	$\pm 3,20E-05$	0,0023	$\pm 1,70E-05$
10	0,1985	$\pm 1,91E-04$	0,0009	$\pm 1,42E-05$	0,0004	$\pm 1,26E-05$	4,73E-05	$\pm 3,65E-06$
12	0,1133	$\pm 1,80E-04$	1,11E-05	$\pm 2,14E-06$	3,40E-06	$\pm 1,02E-06$	0	0
14	0,0540	$\pm 1,55E-04$	0	0	0	0	0	0

Tabla C.7: Valores de BLER con su respectivo IC para el auto codificador (n,6) normalizado en potencia.

E_b/N_o dB	2,6		6,6		9,6		12,6	
	BLER media	IC 95 %						
0	0,8299	$\pm 2,85E-04$	0,6176	$\pm 4,08E-04$	0,5574	$\pm 3,21E-04$	0,5374	$\pm 4,46E-04$
2	0,7693	$\pm 3,33E-04$	0,4410	$\pm 3,61E-04$	0,3637	$\pm 4,30E-04$	0,3396	$\pm 2,91E-04$
4	0,6873	$\pm 2,82E-04$	0,2447	$\pm 3,59E-04$	0,1716	$\pm 3,07E-04$	0,1504	$\pm 3,09E-04$
6	0,5813	$\pm 2,84E-04$	0,0895	$\pm 2,19E-04$	0,0475	$\pm 9,58E-05$	0,0374	$\pm 9,93E-05$
8	0,4541	$\pm 2,23E-04$	0,0172	$\pm 8,23E-05$	0,0059	$\pm 6,40E-05$	0,0039	$\pm 4,00E-05$
10	0,3183	$\pm 3,38E-04$	0,0013	$\pm 2,02E-05$	2,45E-04	$\pm 1,15E-05$	1,16E-04	$\pm 6,55E-06$
12	0,1930	$\pm 2,07E-04$	2,48E-05	$\pm 3,35E-06$	1,20E-06	$\pm 8,16E-07$	3,00E-07	$\pm 2,99E-07$
14	0,0977	$\pm 1,63E-04$	0	0	0	0	0	0

Tabla C.8: Valores de BLER con su respectivo IC para el auto codificador (n,7) normalizado en potencia.

E_b/N_o dB	2,7		7,7		14,7	
	BLER media	IC 95 %	BLER media	IC 95 %	BLER media	IC 95 %
0	0,8934	$\pm 2,12E-04$	0,6803	$\pm 2,58E-04$	0,5574	$\pm 3,63E-04$
2	0,8520	$\pm 1,71E-04$	0,5053	$\pm 3,43E-04$	0,3637	$\pm 4,62E-04$
4	0,7933	$\pm 3,61E-04$	0,2980	$\pm 3,22E-04$	0,1716	$\pm 2,68E-04$
6	0,7140	$\pm 4,17E-04$	0,1212	$\pm 3,27E-04$	0,0475	$\pm 1,50E-04$
8	0,6102	$\pm 3,28E-04$	0,0282	$\pm 1,06E-04$	0,0059	$\pm 7,17E-05$
10	0,4854	$\pm 2,36E-04$	0,0030	$\pm 3,69E-05$	2,45E-04	$\pm 1,10E-05$
12	0,3510	$\pm 2,19E-04$	1,15E-04	$\pm 4,94E-06$	1,20E-06	$\pm 1,37E-06$
14	0,2250	$\pm 4,01E-04$	1,50E-06	$\pm 8,39E-07$	0	0

Apéndice D

D.1. Valores de BER para auto codificadores (n,k) normalizados en energía.

Tabla D.1: Valores de BER para el auto codificador (n,3) normalizado en energía.

E_b/N_0	(2,3)		(3,3)		(4,3)		(6,3)	
	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %
0	0,294442	0,00013732	0,15615943	0,00016023	0,10534272	8,97E-05	0,06781203	7,73E-05
2	0,227542	0,00027034	0,10124842	8,86E-05	0,06285547	5,62E-05	0,03917065	4,55E-05
4	0,159783	0,00014869	0,05387905	1,22E-04	0,02920178	3,85E-05	0,01735099	2,04E-05
6	0,098447	9,98E-05	0,02104224	2,54E-05	0,00899833	2,63E-05	0,00505244	2,11E-05
8	0,049853	7,62E-05	0,00511856	2,19E-05	0,00142961	1,45E-05	0,00074268	4,92E-06
10	0,018825	5,28E-05	0,00058	9,67E-06	7,79E-05	2,76E-06	3,98E-05	1,74E-06
12	0,004508	3,01E-05	2,15E-05	2,67E-06	8,25E-07	2,54E-07	4,67E-07	9,49E-08
14	0,000533	1,14E-05	1,67E-07	1,09E-07	0	0	0	0

Tabla D.2: Valores de BER para el auto codificador (n,4) normalizado en energía.

E_b/N_0	2,4		4,4		6,4		8,4	
	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %
0	0.44844557	$\pm 2.1676\text{E-}04$	0.152654985	± 0.00015012	0.08989116	$\pm 7.13\text{E-}05$	0.06252875	$\pm 5.39\text{E-}05$
2	0.38707437	$\pm 1.7631\text{E-}04$	0.097107575	± 0.00011134	0.05198439	$\pm 6.25\text{E-}05$	0.03407432	$\pm 5.60\text{E-}05$
4	0.32066431	$\pm 2.2115\text{E-}04$	0.049923711	$\pm 8.12\text{E-}05$	0.0228474	$\pm 3.60\text{E-}05$	0.01349384	$\pm 3.44\text{E-}05$
6	0.25229201	$\pm 2.6111\text{E-}04$	0.018294148	$\pm 3.45\text{E-}05$	0.00638673	$\pm 2.14\text{E-}05$	0.00314086	$\pm 1.22\text{E-}05$
8	0.18406875	$\pm 2.0972\text{E-}04$	0.003911996	$\pm 1.31\text{E-}05$	0.0008719	$\pm 7.45\text{E-}06$	0.00030467	$\pm 2.00\text{E-}06$
10	0.12127225	$\pm 1.3113\text{E-}04$	3.67252E-04	$\pm 5.57\text{E-}06$	3.78E-05	$\pm 1.34\text{E-}06$	7.25E-06	$\pm 4.97\text{E-}07$
12	0.06923238	$\pm 1.2354\text{E-}04$	1.04E-05	$\pm 9.80\text{E-}07$	1.83E-07	$\pm 1.42\text{E-}07$	2.50E-08	$\pm 3.27\text{E-}08$
14	0.03240976	$\pm 9.29\text{E-}05$	5.00E-08	$\pm 6.53\text{E-}08$	0	± 0	0	± 0

Tabla D.3: Valores de BER para el auto codificador (n,5) normalizado en energía.

E_b/N_0	(2,5)		(5,5)		(6,5)		(10,5)	
	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %
0	0,58529556	3,7374E-04	0,147989114	0,00012361	0,11585391	8,72E-05	0,06099471	3,51E-05
2	0,5375244	2,2835E-04	0,092335002	0,00013267	0,06887965	7,15E-05	0,03274294	4,78E-05
4	0,48542126	2,2478E-04	0,045306674	5,04E-05	0,031405	3,96E-05	0,01257103	2,68E-05
6	0,42971353	2,4184E-04	0,015177137	3,28E-05	0,00929338	2,37E-05	0,0027369	1,18E-05
8	0,37221127	2,1867E-04	0,002705318	1,06E-05	0,00134332	5,66E-06	0,00023307	2,85E-06
10	0,31486769	3,0029E-04	0,000177623	3,17E-06	6,89E-05	8,95E-07	4,71E-06	4,24E-07
12	0,26139119	2,3004E-04	2,08E-06	3,75E-07	6,67E-07	1,62E-07	2,00E-08	2,61E-08
14	0,21536128	1,9137E-04	2,00E-08	3,92E-08	0	0	0	0

D.2. Valores de BER para auto codificadores (n,k) normalizados en potencia.

Tabla D.4: Valores de BER con su respectivo IC para el auto codificador (n,2) normalizado en potencia.

E_b/N_o dB	2,2		3,2		4,2	
	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %
0	0,0791	$\pm 4,41E-05$	0,0425	$\pm 9,68E-05$	0,0317	$\pm 5,09E-05$
2	0,038	$\pm 1,34E-04$	0,0173	$\pm 6,08E-05$	0,013	$\pm 2,51E-05$
4	0,0129	$\pm 5,03E-05$	0,0044	$\pm 2,42E-05$	0,0034	$\pm 1,94E-05$
6	0,0025	$\pm 1,36E-05$	5,48E-04	$\pm 8,06E-06$	4,24E-04	$\pm 6,06E-01$
8	2,24E-04	$\pm 7,12E-06$	2,13E-05	$\pm 1,29E-06$	1,81E-05	$\pm 1,66E-06$
10	5,65E-06	$\pm 1,00E-06$	1,33E-07	$\pm 1,07E-07$	1,25E-07	$\pm 8,8E-08$
12	5,00E-08	$\pm 9,80E-08$	0	± 0	0	± 0
14	0	± 0	0	± 0	0	± 0

Tabla D.5: Valores de BER con su respectivo IC para el auto codificador (n,4) normalizado en potencia.

E_b/N_o	2,4		4,4		6,4		8,4	
	dB	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %	BER
0	0,36909	$\pm 1,19E-04$	0,1520	$\pm 1,14E-04$	0,0908	$\pm 9,29E-05$	0,0653	$\pm 6,75E-05$
2	0,29497	$\pm 1,93E-04$	0,0966	$\pm 1,08E-04$	0,0528	$\pm 8,25E-05$	0,0365	$\pm 3,62E-05$
4	0,21972	$\pm 2,43E-04$	0,0494	$\pm 7,61E-05$	0,0234	$\pm 2,56E-05$	0,015	$\pm 3,28E-05$
6	0,15138	$\pm 1,37E-04$	0,0179	$\pm 3,69E-05$	0,0066	$\pm 1,89E-05$	3,80E-03	$\pm 1,56E-05$
8	0,09615	$\pm 1,37E-04$	0,0037	$\pm 1,33E-05$	9,29E-04	$\pm 3,69E-06$	4,23E-04	$\pm 3,00E-06$
10	0,05698	$\pm 7,58E-05$	0,0003	$\pm 5,34E-06$	4,16E-05	$\pm 9,49E-07$	1,31E-05	$\pm 8,46E-07$
12	0,03099	$\pm 4,38E-05$	7,03E-06	$\pm 5,39E-07$	3,33E-07	$\pm 1,09E-07$	5,00E-08	$\pm 4,00E-08$
14	0,01495	$\pm 6,61E-05$	2,50E-08	$\pm 4,90E-08$	0	± 0	0	± 0

Tabla D.6: Valores de BER con su respectivo IC para el auto codificador (n,5) normalizado en potencia.

E_b/N_o	2,5		5,5		6,5		10,5	
	dB	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %	BER
0	0,4752	$\pm 2,42E-04$	0,1803	$\pm 1,17E-04$	0,1148	$\pm 7,25E-05$	0,0753	$\pm 6,43E-05$
2	0,4013	$\pm 2,40E-04$	0,1127	$\pm 1,01E-04$	0,0679	$\pm 5,08E-05$	0,0404	$\pm 3,30E-05$
4	0,3211	$\pm 1,91E-04$	0,0555	$\pm 8,13E-05$	0,0307	$\pm 3,86E-05$	0,0155	$\pm 2,17E-05$
6	0,2403	$\pm 2,49E-04$	0,0185	$\pm 5,73E-05$	0,0089	$\pm 2,32E-05$	0,0034	$\pm 1,08E-05$
8	0,1663	$\pm 2,29E-04$	0,0033	$\pm 8,57E-06$	0,0013	$\pm 5,37E-06$	0,0003	$\pm 2,13E-06$
10	0,1047	$\pm 1,07E-04$	0,0002	$\pm 3,56E-06$	6,00E-05	$\pm 2,09E-06$	5,91E-06	$\pm 4,56E-07$
12	0,0584	$\pm 9,58E-05$	2,78E-06	$\pm 5,34E-07$	5,67E-07	$\pm 1,70E-07$	0	0
14	0,0274	$\pm 7,97E-05$	0	0	0	0	0	0

Tabla D.7: Valores de BER con su respectivo IC para el auto codificador (n,6) normalizado en potencia.

E_b/N_o	2,6		6,6		9,6		12,6	
	dB	BER	IC 95 %	BER	IC 95 %	BER	IC 95 %	BER
0	0,5876	$\pm 3,46E-04$	0,2136	$\pm 2,09E-04$	0,1270	$\pm 1,06E-04$	0,0919	$\pm 1,10E-04$
2	0,5197	$\pm 3,46E-04$	0,1353	$\pm 1,40E-04$	0,0726	$\pm 1,05E-04$	0,0505	$\pm 5,23E-05$
4	0,4408	$\pm 2,52E-04$	0,0678	$\pm 1,11E-04$	0,0309	$\pm 5,98E-05$	0,0202	$\pm 4,46E-05$
6	0,3530	$\pm 2,19E-04$	0,0232	$\pm 5,87E-05$	0,0081	$\pm 1,66E-05$	0,0048	$\pm 1,28E-05$
8	0,2611	$\pm 1,51E-04$	0,0043	$\pm 2,08E-05$	0,0010	$\pm 1,07E-05$	4,83E-04	$\pm 5,02E-06$
10	0,1744	$\pm 2,05E-04$	3,15E-04	$\pm 5,05E-06$	4,08E-05	$\pm 1,91E-06$	1,45E-05	$\pm 8,19E-07$
12	0,1017	$\pm 1,15E-04$	6,20E-06	$\pm 8,39E-07$	2,00E-07	$\pm 1,36E-07$	3,75E-08	$\pm 3,74E-08$
14	0,0501	$\pm 8,57E-05$	0	0	0	0	0	0

Tabla D.8: Valores de BER con su respectivo IC para el auto codificador (n,7) normalizado en potencia.

E_b/N_o	2,7		7,7		14,7	
	dB	BER	IC 95 %	BER	IC 95 %	BER
0	0,6735	$\pm 3,24E-04$	0,3162	$\pm 1,84E-04$	0,2136	$\pm 1,87E-04$
2	0,6153	$\pm 2,22E-04$	0,2091	$\pm 1,83E-04$	0,1272	$\pm 1,74E-04$
4	0,5453	$\pm 3,97E-04$	0,1113	$\pm 1,36E-04$	0,0575	$\pm 7,99E-05$
6	0,4652	$\pm 3,89E-04$	0,0421	$\pm 1,19E-04$	0,0166	$\pm 3,95E-05$
8	0,3756	$\pm 2,63E-04$	0,0095	$\pm 3,61E-05$	0,0023	$\pm 1,80E-05$
10	0,2826	$\pm 1,65E-04$	0,0010	$\pm 1,23E-05$	1,11E-04	$\pm 2,74E-06$
12	0,1944	$\pm 1,36E-04$	3,82E-05	$\pm 1,65E-06$	1,08E-06	$\pm 3,43E-07$
14	0,1197	$\pm 2,28E-04$	5,00E-07	$\pm 2,80E-07$	0	0

Referencias

- [1] P. E. Montero, "Aprendizaje por refuerzo en espacios continuos," Ph.D. dissertation, Tesis de doctorado, Universidad de Valencia, Espana, 2014.
- [2] X. Hao, G. Zhang, y S. Ma, "Deep learning," *International Journal of Semantic Computing*, vol. 10, num. 03, pp. 417–439, 2016.
- [3] L. Noriega, "Multilayer perceptron tutorial," *School of Computing. Staffordshire University*, vol. 4, num. 5, p. 444, 2005.
- [4] Jesus, "Gráfica completa de redes neuronales," 2022. [En línea]. Disponible: <https://www.datasmarts.net/grafica-completa-de-redes-neuronales/>
- [5] L. V. Bermeo Chimbo, "Análisis de eliminación de interferencias en arreglos de antenas lineales inteligentes basados en algoritmos adaptativos de gradiente estocástico," B.S. thesis, 2015.
- [6] T. O'shea y J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, num. 4, pp. 563–575, 2017.
- [7] S. Rosales Magallares, "Machine learning para comunicaciones," B.S. thesis, 2019.
- [8] P. Saraiva, "On shannon entropy and its applications," *Kuwait Journal of Science*, 2023.
- [9] J. L. Sarmiento-Ramos, "Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica," *Revista UIS Ingenierías*, vol. 19, num. 4, pp. 1–18, 2020.
- [10] P. Li, Y. Pei, y J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Applied Soft Computing*, p. 110176, 2023.
- [11] D. Bank, N. Koenigstein, y R. Giryes, "Autoencoders," *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.
- [12] F. J. Marchal Cebador, "Auto-codificadores para detección y procesado de imágenes," 2019.
- [13] T. J. O'Shea, K. Karra, y T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2016, pp. 223–228.

- [14] M. E. Morocho-Cayamcela, J. N. Njoku, J. Park, y W. Lim, "Learning to communicate with autoencoders: Rethinking wireless systems with deep learning," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAICC)*. IEEE, 2020, pp. 308–311.
- [15] N. Rajapaksha, N. Rajatheva, y M. Latva-aho, "Low complexity autoencoder based end-to-end learning of coded communications systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–7.
- [16] M. Goutay, F. A. Aoudia, y J. Hoydis, "Deep reinforcement learning autoencoder with noisy feedback," in *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*. IEEE, 2019, pp. 1–6.
- [17] J. Wang, W. Wang, F. Luo, y S. Wei, "Modulation classification based on denoising autoencoder and convolutional neural network with gnu radio," *The Journal of Engineering*, vol. 2019, num. 19, pp. 6188–6191, 2019.
- [18] S. Dörner, S. Cammerer, J. Hoydis, y S. Ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, num. 1, pp. 132–143, 2017.
- [19] S. GALLARDO VÁZQUEZ, *Elementos de sistemas de telecomunicaciones 2*. Ediciones Paraninfo, SA, 2019.
- [20] B. Sklar, *Digital communications: fundamentals and applications*. Pearson, 2021.
- [21] R. Wiesmayr, G. Marti, C. Dick, H. Song, y C. Studer, "Bit error and block error rate training for ml-assisted communication," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.
- [22] L. Rouhiainen, "Inteligencia artificial," *Madrid: Alienta Editorial*, pp. 20–21, 2018.
- [23] A. López Mora, "Reducción de ruido en señales de audio basada en una red neuronal convolucional," 2019.
- [24] P. P. Shinde y S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018.
- [25] J. M. Gata Romero, "Autoencoders variacionales para la detección de ataques en redes de telecomunicaciones," 2021.

- [26] D. Goikoetxea Pardo, “Entrenamiento evolutivo de autoencoders,” Ph.D. dissertation, ETSI_Informatica, 2018.
- [27] A. E. Repetur, “Redes neuronales artificiales,” *Buenos Aires: Universidad Nacional del Centro de la Provincia de Buenos Aires*, 2019.
- [28] A. Vilagran Solsona, “Facial expression detection using convolutional neural networks,” B.S. thesis, Universitat Politècnica de Catalunya, 2018.
- [29] J. Sánchez y M. A. Campos, “Red neuronal artificial para detección de armas de fuego y armas blancas en video vigilancia,” *Revista de Iniciación Científica*, vol. 7, num. 2, pp. 83–88, 2021.
- [30] A. I. Bonilla, “Machine learning: Redes neuronales aplicadas a la fórmula 1,” 6 2023.
- [31] J. D. Ramírez Sánchez, “Regularización de redes neuronales artificiales para la clasificación de imágenes de retinopatía diabética,” Ph.D. dissertation, Universidad Nacional de Colombia.
- [32] S. Shen, Z. Yao, A. Gholami, M. Mahoney, y K. Keutzer, “Powernorm: Rethinking batch normalization in transformers,” in *International conference on machine learning*. PMLR, 2020, pp. 8741–8751.
- [33] A. Dorado Valín, “Análisis del impacto de las medidas de distancia en técnicas de reducción de la dimensionalidad,” 2023.
- [34] S. Carmona Aguiar, “Generación de información acústica sintética usando redes neuronales: variational autoencoder y conditional variational autoencoder,” 2023.
- [35] F. Serra Bisquerra, “Técnicas de aprendizaje automático para evaluar el riesgo de cristalización del ácido úrico.”
- [36] H. Y. S, B. T. S, S. G. R, R. K. K. N, y V. Karjigi, “Wireless communication using autoencoder,” in *2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSS)*, 2023, pp. 1–6.
- [37] Zaruma26, “Ae_coding,” https://github.com/Zaruma26/AE_coding.git, 2024, accessed: 2024-06-21.
- [38] M. E. Morocho-Cayamcela y W. Lim, “Accelerating wireless channel autoencoders for short coherence-time communications,” *Journal of Communications and Networks*, vol. 22, num. 3, pp. 215–222, 2020.