Estimación de la OSNR Usando Histogramas 2D y Aprendizaje Profundo en Sistemas WDM sin Rejilla

Kevin D. Martinez Zapata[1]

[1]*Universidad de Antioquia, Facultad de Ingeniería, GITA lab., calle 67 No. 53 – 108, Medellín, Colombia*

kevin.martinez1@udea.edu.co

1. Presentación

Este **resumen ejecutivo** da cuenta del proceso de desarrollo de estrategias de aprendizaje profundo para resolver un problema de estimación de la OSNR en sistemas WDM sin rejilla modulado en 16-QAM con una tasa de transmisión de 16 GBd en el marco del curso de **Deep Learning** de la Universidad de Antioquia durante el año 2024. Este documento se construyó con las directrices establecidas por el profesor responsable del curso.

El objetivo de este documento es presentar las diferentes estrategias para solucionar un problema del área de las comunicaciones ópticas usando histogramas en 2D y tres diferentes arquitecturas de redes neuronales profundas. Asimismo, mostrar la estructura de los notebooks diseñados para solución del problema, como se abordó el problema y los resultados obtenidos.

El contenido de este documento está organizado en 6 secciones principales, según los requerimientos anteriormente mencionados, incluyendo la sección de presentación: La sección 2 se describe cómo están organizados los notebooks proporcionados para facilitar su comprensión y replicación. La sección 3 muestra las fases de la solución, divididas en preprocesamiento de los datos, exploración de los datos, entrenamiento del modelo y validación. La sección 4 explica el proceso iterativo sobre los modelos presentados, como la variación de los parámetros. Finalmente, las secciones 5 y 6 presentan los resultados de los modelos de aprendizaje profundo y las conclusiones del trabajo, respectivamente.

2. Descripción de la estructura de los notebooks entregados

Para el desarrollo de la solución planteada, se propusieron 4 notebooks de Jupyter claramente diferenciados entre sí, siendo de ellos dedicados exclusivamente a cada una de las arquitecturas propuestas. Además, un solo notebook para la exploración de los datos. Los notebooks 02 al 04

son similares, ya que todos inician cargando los datos y haciendo las respectivas **normalizaciones** y **reshape** a los datos, siendo el modelo lo diferente. En cada uno de ellos se hace proceso de validación cruzada y se obtiene el resultado para un conjunto de parámetros específicos. En los resultados se mostrarán los resultados para diferentes combinaciones de parámetros de la red.

a. 01 – exploración de los datos

En este notebook se ofrece una vista previa de los datos que se van a usar para entrenar y validar los modelos de red neuronal profunda que serán presentados mas adelante. Los datos son descargados directamente al entorno de trabajo. Los datos son 729 imágenes de histogramas en dos dimensiones de los diagramas de fase y cuadratura de símbolos transmitidos a través de fibra óptica y modulados en 16-QAM. A continuación, la figura 1 muestra algunos de los escenarios que hay contemplados en la base de datos.

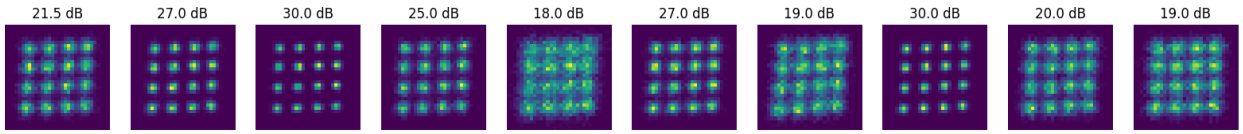

Fig. 1. Histogramas 2D de los diagramas de fase y cuadratura.

En este caso, el proceso de etiquetado de los datos se hizo de forma automática al cargar los datos, ya que se sabe cual es el nivel de relación señal a ruido óptico (OSNR) de cada una de las constelaciones como se muestra en la imagen. Adicionalmente, se cuenta con una característica extra que da información del espaciamiento de canal de cada una de las imágenes.

Las imágenes cargadas están en un rango de 0 hasta 255 con un solo canal (no RGB). Por otro lado, se etiquetaron las imágenes, en este caso, usando el valor de OSNR como la variable que se quiere predecir o estimar. Finalmente, la característica adicional en el modelo es el espaciamiento de canal, que inicialmente es una variable categórica, pero que posteriormente se transforma a una variable codificada usando **one-hot encoding**, de forma que los modelos de red neuronal profunda entiendan esta información.

b. 02 – arquitectura de línea base

La arquitectura de línea base es un modelo de red neuronal simplemente densa, que tiene como objetivo hacer un primer acercamiento a una respuesta del problema. Se usa este modelo ya que

es el mas simple y menos complejo que los que se desarrollaron más adelante. El modelo simplemente denso tiene la estructura mostrada en la figura 2. Para implementar este modelo, es necesario aplanar las imágenes, convirtiendo cada imagen en un vector de 1024 valores – Esto porque las imágenes son de 32 x 32 pixeles – para que la red pueda procesarlos de forma correcta.

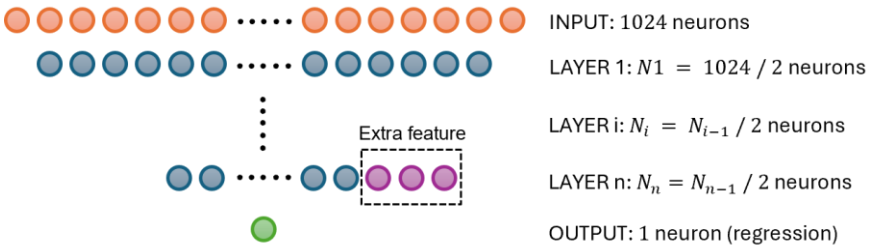

Fig. 2. Modelo de red neuronal densa.

En esta arquitectura, se tiene una capa de entrada de 1024 neuronas y una capa de salida de una neurona son una función de activación lineal para regresión. Las capas ocultas de la red están distribuidas de forma que cada una de las capas ocultas, desde las más cercana a la capa de salida hasta las mas alejada, tienen en total la mitad de las neuronas de la capa anterior, hasta tener la cantidad de capas densas deseadas. En este caso, se llegó hasta 32 neuronas para un total de 5 capas ocultas. La función de activación de las capas ocultas está determinada por un parámetro dentro de la función de crea el modelo, por defecto se una la función de activación ReLU para evitar activaciones negativas y desvanecimiento del gradiente. Además, los valores de las entradas están normalizados entre 0 y 1 usando un escalador mínimo-máximo, por lo que no tiene sentido usar una función de activación que este definida para valores negativos. Finalmente, la función que crea el modelo tiene la posibilidad de añadir regularización L2 en las dos primeras capas ocultas para evitar el sobreajuste.

c. 03 – autoencoder con regresor

Esta propuesta es basada en idea de que los autoencoder tienen la capacidad de extraer las características más relevantes de un conjunto de datos comprimiendo la información. Este proceso implicar perder información, en este caso, de las imágenes, pero mantiene lo mas importante. La figura 3 muestra la arquitectura del autoencoder con una capa de entrada de 1024 neuronas, una capa **encoder** que tiene una cantidad de neuronas determinada por el parámetro *code_size* dentro de la función que crea el modelo, y una capa de salida llamada **decoder** que trata de reconstruir la imagen original a partir de esa representación comprimida.

Fig. 3. Arquitectura de autoencoder.

En el contexto de las constelaciones, la figura 4 muestra las imágenes antes y después de pasar por el autoencoder, evidenciándose una pérdida de información. Sin embargo, la dispersión de los datos y las regiones más densas dentro de la constelación se mantienen, pero ahora en una representación menos ruidosa debido a que es reconstruida de una representación comprimida de los datos, lo que puede ayudar al regresor mostrado en la figura 2 a converger más rápido y reducir el tiempo de la etapa de entrenamiento.

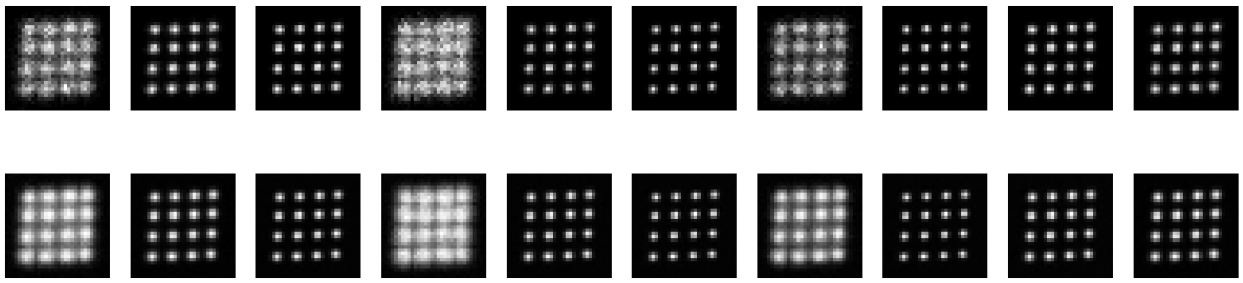

Fig. 4. Imágenes originales vs imágenes de salida del autoencoder.

El objetivo con esta arquitectura es obtener un modelo intermedio con la capa **encoder** que es la que contiene la información comprimida de las imágenes y usar esas representaciones como entrada para el regresor usando anteriormente para comparar si incluir esta etapa previa de extracción de características genera un efecto positivo en la estimación.

d. 04 – red neuronal convolucional

Ya que el problema que se quiere abordar es con imágenes, tiene más sentido usar una red neuronal convolucional (CNN), ya que es una arquitectura de red que está diseñada para detectar patrones locales dentro de una imagen por medio de filtros. Similar a los demás notebooks, en este se propone una arquitectura de CNN como la que se muestra en la figura 5, con 3 capas convolucionales de 4, 8 y 16 filtros, respectivamente, de dimensiones 3x3. Asimismo, se usó la técnica de relleno de ceros a los bordes y capas de MaxPooling en medio de dos capas convolucionales para reducir la espacialidad de las imágenes sin afectar directamente a los pesos de los filtros.

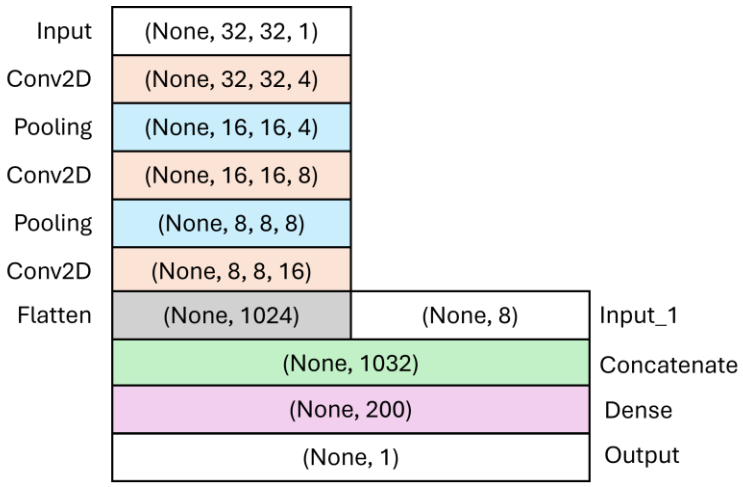

Fig. 5. Arquitectura de la CNN.

Para esta arquitectura, se usó la función de activación ReLU al igual que las arquitecturas anteriores. Además, se tiene la particularidad que, en este caso, la característica adicional de espaciamiento se agrega luego del proceso convolucional de la red, o sea, en la parte completamente conectada de la red, luego de hacer el aplanado. Finalmente, se tiene una capa de concatenación que junta la característica adicional con las características extraídas de la capa convolucional. Luego una capa densa de 200 neuronas con activación ReLU, y, por último, la capa de salida con una sola neurona y con activación lineal.

3. Descripción de la solución

La solución fue diseñada en 4 pasos fundamentales que son: i) Preprocesamiento de los datos; ii) Exploración de los datos; iii) entrenamiento de los modelos; iv) Evaluación de los modelos. En la fase i) se hizo la transformación de las imágenes a usando un filtro gaussiano de kernel 5x5 para filtrar la imagen y resaltar las regiones mas densas. Se identificaron además las características mas importantes dentro de la imagen como lo son los bordes, que dan cuenta de la cantidad de distorsión de las constelaciones, y el centroide de estas que dan cuenta del nivel de desviación del punto ideal producto de las distorsiones en la fibra. En la fase ii) se hizo el etiquetado de los datos tanto de la variable respuesta como de la característica adicional de espaciamiento de canal. Finalmente, en las fases iii) y iv) se entrenaron los 3 modelos de DNN mostrados en la sección 2 usando el proceso de validación cruzada para garantizar que el modelo es independiente de la partición de los datos que se haga.

La herramienta utilizada para el desarrollo de este trabajo fue Python, más específicamente los notebooks de Jupyter en conjunto con algunas librerías especializadas como Tensorflow, scikit-learn, Pandas, entre otras. Adicionalmente, hay un par de funciones dentro del espacio de trabajo en un archivo de tipo .py donde se encuentras funciones para cargar los datos de la forma (imágenes, etiquetas, característica extra). Tensorflow es usado para crear los modelos de DNN y entrenarlos. Mientras que scikit-learn es usada para todo el tema de la evaluación de los modelos, como la partición de los datos para la validación cruzada.

La solución consiste, exactamente, en estimar el valor de OSNR o nivel de ruido por el que está afectado una constelación, pudiendo usar o no, información del espaciamiento de canal que corresponde a esa constelación en particular. Los resultados de este problema son dados en decibeles (dB), siendo que la mínima separación (granularidad) es de 1 dB.

4. Descripción de las iteraciones realizadas

En la primera iteración, se hizo una normalización de las imágenes para que estén en un rango de 0 a 1 y se diseñó un modelo base que corresponde a una red puramente densa con 5 capas ocultas para probar la capacidad de la red de estimar la OSNR. En este primer acercamiento, el error fue aparentemente muy bueno, lo cual es bastante extraño conociendo el fenómeno.

En la segunda iteración, se hizo un análisis detallado de los datos que entraban a la red y los valores de que entregaba. En este punto se pudo observar que los valores que se quieren predecir, que estaban escalados usando el mínimo y máximo, no se estaban desescalando cuando se hacia la validación, por lo que los resultados de la red densa son resultados escalados.

En la tercera iteración se modificó el escalamiento de los valores a predecir usando un escalador mínimo-máximo ofrecido por sckit-learn para mas facilidad, verificando esta vez que se calculara el error usando el valor original y no escalado. En este caso, los resultados fueron más realistas a los anteriores, pero seguían siendo altos para la granularidad de los datos.

En la cuarta iteración, se hicieron modificaciones a la red, añadiendo un número mayor de épocas al entrenamiento, capas de regularización y el proceso de validación cruzada usando particiones aleatorias, y no una sola partición fija. Esta iteración mejoro significativamente el rendimiento de la red, por lo que se procedió a verificar con los demás modelos.

En la quinta iteración, se diseñaron los demás modelos usando el mismo preprocesado de los datos y la misma función que hace la validación cruzada, solamente variando el modelo de DNN, obteniendo en los tres casos valores relativamente buenos.

En la sexta y última iteración, se hizo el proceso de variar los parámetros de los tres modelos para obtener resultados para diferentes arquitecturas y poder compararlas, obteniendo el mejor conjunto de parámetros, según el enfoque, para este problema en particular.

5. Descripción de los resultados.

Los resultados obtenidos para este trabajo están en términos de los dB, y fueron estimados usando la función de perdida de error absoluto medio (MAE), que entrega los resultados en las mismas unidades que la variable original que se quiere estimar y no penaliza tanto los valores atípicos.

a. Resultados de la red puramente densa

La tabla 1 muestra los resultados de la red neuronal densa para diferente numero de capas y neuronas. El número de neuronas varias desde 2^7 hasta 2^{10}, siendo que el numero de neuronas en la última capa oculta debe ser de 32. Por esta razón, el numero de capas varías desde 3 hasta 6. Adicionalmente, parar cada conjunto de parámetros, se entrenó un modelo sin considerar la característica adicional y considerándola. Para este modelo, los mejores resultados en términos de los dB son cuando hay 512 (5 capas ocultas) incluyendo la información adicional, obteniendo un error de 0.82 dB. Por otro lado, el peor rendimiento fue de 1.06 dB cuando se tienen 6 capas ocultas iniciando con una capa de 1024 neuronas. Sin embargo, al incluir la información de canal se reduce a menos de 1 dB, que es lo que se desea.

Tabla 1. Resultados de la red densa

Nro. de neuronas	Información del espaciamiento de canal	MAE [dB]
1024	Yes	0.88 +/- 0.17
	No	1.06 +/- 0.43
512	Yes	0.82 +/- 0.17
	No	0.95 +/- 0.22

256	Yes	0.82 +/- 0.20
	No	1.02 +/- 0.24
128	Yes	0.91 +/- 0.18
	No	1.15 +/- 0.24

b. Resultados para el enfoque con autoencoder

La tabla 2 muestra los resultados de usar un autoencoder para extraer las principales características de las imágenes para meterla a la red densa del literal anterior. Los resultados son basados en el mejor modelo de la red densa, que es el que tiene 5 capas ocultas desde 512 hasta 32 neuronas. El parámetro que varia en este caso es el tamaño de código del autoencoder. En el mejor caso se obtuvo un error de 0.8 dB, incluyendo la información de canal, usando un tamaño de código de 128, lo que significa una reducción del espacio en 8 veces. Por otro lado, usar un tamaño de código de 512 da ligeramente peor que el caso anterior habiendo comprimido la información a solo la mitad. Esto puede indicar que aún en ese nivel quedan características basura que confunden un poco a la red. Finalmente, era de esperarse que si se comprime demasiado la información, hasta 32, el error aumentaría significativamente con respecto a los demás.

Tabla 2. Resultados de la red densa con autoencoder.

Tamaño de código	Información del espaciamiento de canal	MAE [dB]
32	Yes	1.83 +/- 0.22
	No	3.09 +/- 0.37
128	Yes	0.80 +/- 0.17
	No	0.96 +/- 0.20
512	Yes	0.94 +/- 0.14
	No	1.10 +/- 0.16

c. Resultados de la CNN

La table 3 muestra los resultados de la CNN con 3 capas convolucionales. En el primer caso, las capas convolucionales están conformadas por 4, 8, y 16 filtros obteniendo un error de 1.04 dB

en la estimación, incluyendo la información del canal. Mientras que, al aumentar el número de filtros por cada capa, el error aumenta ligeramente.

Filters	Información del espaciamiento de canal	MAE [dB]
(4, 8, 16)	Yes	1.04 dB +/- 0.18
	No	1.21 dB +/- 0.12
(16, 32, 64)	Yes	1.13 dB +/- 0.17
	No	1.34 dB +/- 0.19

6. Conclusiones

Se propusieron tres estrategias de estimación de la OSNR en sistemas WDM sin rejilla modulados en 16-QAM con una tasa de transmisión de 16 GBd usando histogramas en 2D en conjunto con diferentes arquitecturas de DNNs. Estos métodos contribuyen con la diferenciación entre ICI lineal y OSNR debido al ruido de emisión espontanea amplificada. Estos métodos lograron un error inferior a los 2 dB en la estimación de la OSNR sin incluir la información del canal en canales espectralmente traslapados (banda de guarda < 0 GHz). Esto es especialmente beneficioso, ya que permite la estimación ciega de la OSNR en futuras redes ópticas sin rejilla que sólo procesen los símbolos recibidos.