Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Manejo autónomo de un vehículo mediante el uso de Aprendizaje de Máquina

Tesis presentada al

Posgrado en Física Aplicada

Como requisito parcial para obtener el grado de:

Maestro en Ciencias: Física Aplicada

por

Pérez Lima Dalí del Ángel

Asesorado por

Dr. Castillo Mixcóatl Juan

Puebla Pue.

Diciembre del 2023

Manejo autónomo de un vehículo mediante el uso de Aprendizaje de Máquina

Dalí del Ángel Pérez Lima

Dr. Juan Castillo Mixcóatl

B U A P

Título: Manejo autónomo de un vehículo mediante el uso de Aprendizaje de Máquina

Tesista: Pérez Lima Dalí del Ángel

Comité:

_____

Dr. Muñoz Aguirre Severino

Presidente.

_____

Dra. Beltrán Pérez Georgina

Secretaria.

_____

Dr. Hernández López Javier Miguel

Vocal.

_____

Dr. Velázquez Castro Jorge

Suplente.

_____

Dr. Castillo Mixcóatl Juan

Asesor.

AGRADECIMIENTOS

En primer lugar, quiero expresar mi más profundo agradecimiento a mi asesor, el Dr. Juan, por su valiosa orientación, apoyo y paciencia durante el desarrollo de esta investigación. El han sido una fuente de inspiración, motivación y conocimiento, y me han enseñado a ser un mejor investigador y persona.

También quiero agradecer al comité evaluador, integrado por el Dr. Severino, la Dra. Georgina, el Dr. Javier y el Dr. Jorge, por sus atentas observaciones, críticas constructivas y sugerencias pertinentes, que me han permitido mejorar la calidad y el rigor de este trabajo.

Asimismo, quiero agradecer a mi familia, especialmente a mis padres, por su amor incondicional, comprensión y ánimo en todo momento. Ellos han sido el pilar fundamental de mi vida y me han brindado todo lo necesario para alcanzar mis metas. Sin ellos, nada de esto hubiera sido posible.

Además, quiero reconocer el apoyo económico otorgado por el Consejo Nacional de Humanidades, Ciencia y Tecnología (CONACYT), a través del programa de becas para estudios de posgrado. Este apoyo ha sido esencial para la realización de este trabajo y para mi formación académica.

Finalmente, quiero agradecer a todos mis amigos, que me han acompañado en este camino con su amistad, solidaridad y consejo. Ellos han sido un apoyo moral y emocional, y me han hecho más llevadero el proceso de investigación. A todos ellos, mi más sincero reconocimiento y gratitud

Contenido

Agradecimientos v

Contenido vii

Resumen ix

Introducción x

Objetivo Principal xi

Antecedentes xii

Capítulo 1. Fundamentos teóricos 1

1.1 Aprendizaje de Máquina 1

1.1.1 Tipos de Aprendizaje de Máquina *3*

1.1.2 Estructura de una red neuronal *4*

1.1.3 Regresión *6*

1.1.4 Validación y validación cruzada *14*

1.1.5 Curvas de aprendizaje *15*

1.2 Aprendizaje profundo 17

1.2.1 Red neuronal convolucional *18*

1.3 Análisis de Componentes principales (PCA) 20

1.3.1 Regresión de Componentes Principales (PCR) *22*

1.4 Proyección a Estructuras Latentes (PLS) 22

Capítulo 2. Desarrollo experimental 27

2.1 Computadora de placa única 27

2.1.1 Raspberry Pi 4 ..... *27*

2.1.2 Jetson nano *28*

2.1.3 Mini PC MELE *28*

2.2 Arreglo experimental 30

2.2.1 Primer Prototipo *30*

2.2.2 Segundo Prototipo *31*

2.2.3 Tercer Prototipo *34*

2.2.4 Cuarto prototipo *39*

Capítulo 3. Resultados Experimentales 41

3.1 Resultados de la implementacion del metodo de PLS-R 42
3.2 Resultados de la implementacion del metodo de PCR..... 50
3.3 Resultados de la implementación de una red neuronal..... 58
CONCLUSIONES 64
BIBLIOGRAFÍA 65

RESUMEN

El uso de vehículos autónomos puede reducir significativamente el número de accidentes viales. Esto se debe a que estos cuentan con un grupo de sensores (cámaras, GPS, LIDAR, radar y sensor IMU) que les permite el acceso a una gran cantidad de información que puede ser usada para un manejo seguro. Como consecuencia de esto, estos vehículos requieren de una computadora potente para el análisis de los datos provenientes de estos [6].

Sin embargo, es posible realizar este proceso con un sistema más simple mediante el uso de distintas técnicas de Aprendizaje de Maquina (Machine Learning, en inglés) y análisis de datos. En este trabajo se propone el desarrollo de un prototipo de vehículo autónomo en su manejo mediante el uso de Aprendizaje de Maquina y algunas técnicas de análisis de datos, para esto solo se necesita una cámara para la adquisición de datos, una computadora para el análisis de estos y una red neuronal entrenada para el manejo del vehículo.

PALABRAS CLAVE:

Inteligencia Artificial, Aprendizaje de Maquina, Aprendizaje Profundo, reconocimiento de patrones, manejo autónomo, Redes Neuronales, Red de Convolución, Regresión de Componentes Principales, Proyección a estructuras latentes.

INTRODUCCIÓN

En este trabajo se proponen diversos modelos para el manejo autónomo de un automóvil usando algoritmos de Aprendizaje de Maquina y análisis de datos, para esto se obtendrá una serie de datos de entrenamiento que contendrán las fotos obtenidas por una cámara y la velocidad del vehículo, estos datos se obtendrán de forma manual, conduciendo el vehículo por diversas pistas de entrenamiento y grabando el recorrido. Estos datos serán procesados por una red neuronal de convolución que formará una serie de filtros digitales que extraerán información relevante de las imágenes obtenidas y reducirán su tamaño, esto con el fin de reducir la carga computacional, posteriormente se usará una red neuronal de regresión que calculará la dirección que debería tener el vehículo con base a las imágenes analizadas, posteriormente, se actualizará el modelo para que la diferencia entre la velocidad obtenida por el modelo y la deseada sea mínima, de esta manera se obtendrá un modelo o hipótesis que servirá para analizar nuevas imágenes y tomar decisiones con respecto a la dirección del vehículo.

Por otra parte, se utilizará un par de técnicas de análisis de datos, la técnica de Regresión de Componentes Principales (PCA, por sus siglas en inglés) y de Proyección a Estructuras Latentes (PLS, por sus siglas en inglés) para analizar los mismos datos de entrenamiento y, por último, se compararán los resultados obtenidos mediante el uso de estas técnicas.

OBJETIVO PRINCIPAL

Desarrollar el prototipo de un vehículo manejado de forma autónoma, mediante el uso de los algoritmos de Aprendizaje de Maquina y de distintas técnicas de análisis de datos para el estudio de las imágenes de su entorno.

OBJETIVOS PARTICULARES

Para lograr el objetivo principal se proponen los siguientes objetivos particulares:

1. Aprender a usar Python para controlar la computadora de tarjeta simple (Single Board Computer, SBC) Raspberry Pi 4.
2. Construcción de un sistema móvil manejado por la SBC Raspberry Pi.
3. Caracterización de los distintos actuadores del prototipo.
4. Estudio de distintas técnicas de Machine Learning, Deep Learning (DL por sus siglas) y análisis de datos.
5. Estudio de distintos módulos de Python para la implementación de algoritmos de ML y DL.
6. Desarrollar una red neuronal para el análisis de datos provenientes de los sensores.
7. Recolección de datos para el entrenamiento.
8. Entrenamiento de los modelos con los datos obtenidos previamente.
9. Evaluación del desempeño de los distintos modelos.

Chen C. [21] utiliza la red neuronal de convolución AlexNet para analizar los datos de entrenamiento obtenidos con el simulador de manejo "The Open Racing Car Simulator" (TORCS), de esta forma se conduce un vehículo anfitrión que mide distintos indicadores y se simula una serie de vehículos para generar tráfico, con este trabajo se logra un error absoluto promedio de la dirección de 1.43°.

De igual forma, Der_Hau Lee [20] propone un modelo de Aprendizaje de Maquina que fusiona Aprendizaje Multitarea (MT), Redes Neuronales de convolución (CNN) y algoritmos de control, en este trabajo se realizan tareas de clasificación y regresión para medir distintos indicadores que se usaron para el manejo del vehículo, los datos de entrenamiento para este modelo fueron obtenidos del simulador de manejo TORCS y Car Learning to Act (CARLA), en este se pueden simular distintos escenarios de manejo, de esta manera se obtienen datos de entrenamiento y de prueba, de esta manera se logra obtener un error absoluto promedio de 1.43° para TORCS y de 1.95° para CARLA.

En los trabajos generados mediante los distintos simuladores se trabaja bajo circunstancias ideales, ya que no se tiene que lidiar con los distintos problemas que lleva el desarrollo de un prototipo real, como lo son el desarrollo mismo del vehículo, los problemas mecánicos a la hora de su manejo y los problemas de control generados por los distintos actuadores. En este trabajo se desarrolla el prototipo de un vehículo a escala para su manejo autónomo.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS

En este capítulo se presentarán los fundamentos teóricos necesarios para entender los conceptos relacionados con Aprendizaje de Máquina, los distintos tipos de Aprendizaje de Máquina y algunas de sus aplicaciones, además se muestran dos técnicas de análisis de datos, el Análisis de Componentes Principales y la Proyección a Estructuras Latentes.

1.1 APRENDIZAJE DE MÁQUINA

El concepto de inteligencia se puede definir como la habilidad de procesar información para tomar decisiones futuras [1, 5], la Inteligencia Artificial es el campo de estudio que imita el aprendizaje humano para la resolución de tareas propias de éste, tales como la clasificación de información, el reconocimiento de patrones en imágenes y el reconocimiento del lenguaje natural, entre otras, esto lo hace mediante el uso de distintos algoritmos y modelos. El Aprendizaje de Máquina es una rama de la Inteligencia Artificial encargada de la formación de estos modelos sin la necesidad de un programador. Existen distintas definiciones de lo que se considera el Aprendizaje de Máquina, pero, la que se puede considerar la más precisa es la siguiente:

> "Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna tarea T y alguna medida de desempeño P, si su desempeño en T, medido por P, mejora con la experiencia E" [1].

La definición anterior, aunque describe bien el concepto, puede parecer un poco confusa, sin embargo, se puede "traducir" de la siguiente manera:

> "Se dice que un programa 'aprende' si a medida que se entrena un modelo con los datos de entrenamiento para alguna tarea, disminuye la diferencia entre la respuesta deseada y la obtenida por el modelo".

El aprendizaje de maquina fue desarrollado para realizar tareas que no pueden ser desempeñadas por la programación tradicional, estas tareas van desde una simple regresión numérica o logística, hasta tareas más complejas como el reconocimiento de patrones en imágenes, este tipo de tareas puede ser desempeñado por casi cualquier ser humano con facilidad, entonces ¿Por qué se utiliza Aprendizaje de Máquina?

La respuesta a esta pregunta es simple, el ser humano es capaz de hacer estas tareas a pequeña escala, pero cuando se trata de un conjunto grande o un conjunto complejo de datos, el Aprendizaje de Maquina es capaz de analizar toda esta información y además puede llegar a tener un mejor desempeño que el de un ser humano.

¿Por qué se utiliza Aprendizaje de Maquina y no programación tradicional para la resolución de este tipo de problemas? Esto se debe a que el Aprendizaje de Maquina puede adaptarse a nuevos datos y un programa tradicional no, si se quisiera analizar algún conjunto de datos para resolver alguna tarea en particular mediante la programación tradicional, este programa solo sería capaz de analizar datos similares a los del conjunto de entrenamiento con el que se desarrolló el programa, por lo que, si se deseara analizar algún dato diferente a los ya analizados, habría que modificar manualmente el programa para que sea capaz de analizar este tipo de datos, mientras que el Aprendizaje de Maquina modifica el modelo de manera automática y de esta manera es capaz de analizar distintos conjuntos de datos.

El aprendizaje de maquina es una técnica que genera un modelo o hipótesis a partir de las características de una serie de datos de entrenamiento, después usa este modelo para analizar nuevos datos y generar una respuesta, este proceso es similar a nuestra forma de aprender, la Figura 1 muestra un esquema del proceso de aprendizaje de esta técnica.

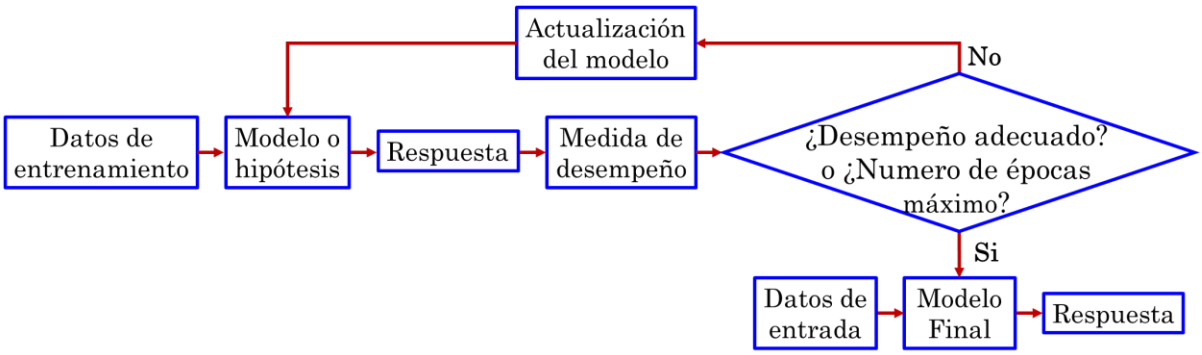


Figura 1. Diagrama de flujo del proceso seguido por un modelo de Aprendizaje de Maquina.

Cabe resaltar que el resultado obtenido mediante Aprendizaje de Máquina será tan bueno o malo como sean los datos de entrenamiento.

1.1.1 Tipos de Aprendizaje de Máquina

Existen tres tipos de Aprendizaje de Máquina, está el Aprendizaje Supervisado, el No Supervisado y el Reforzado [1, 2, 5]. Se utiliza uno u otro dependiendo del tipo de tarea que se desee realizar.

El Aprendizaje Reforzado se usa generalmente en el desarrollo de videojuegos, los datos de entrenamiento para este tipo de aprendizaje están dados de la siguiente forma:

$$D = \{(x_i, y_i, c_i)\}_{i=1}^{N} \quad (1.1)$$

Donde x_i, es la i-ésima entrada, y_i es alguna respuesta para esta entrada, c_i es la calificación para esta respuesta y N es el número total de datos de entrenamiento. En este tipo de aprendizaje se "premian" las respuestas correctas y se "castigan" las respuestas erróneas [1, 2].

Por otra parte, el Aprendizaje No Supervisado se usa generalmente para preprocesar datos, el conjunto de datos de entrenamiento para este tipo de aprendizaje se puede ver de la siguiente manera:

$$D = \{x_i\}_{i=1}^{N} \quad (1.2)$$

Los datos de entrenamiento para este tipo de aprendizaje solo contienen el dato de entrada y el objetivo es que la computadora "descubra" algún patrón o relación entre los datos de este conjunto.

Por último, el Aprendizaje Supervisado se utiliza generalmente para tareas de regresión, regresión logística y clasificación, los datos de entrenamiento para este tipo de aprendizaje están formados de la siguiente forma:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \quad (1.3)$$

Donde x_i es el i-ésimo dato de entrada, y_i es la respuesta deseada para este dato de entrada y N es el número total de datos de entrenamiento, cuando algún dato de entrada es analizado por el modelo, este genera alguna respuesta y el objetivo es hacer que la diferencia entre la salida deseada y la salida obtenida por el modelo sea mínima, una vez que esto sucede, se puede decir que el modelo esta "aprendiendo".

En este trabajo se realizarán tareas de regresión, por lo que se utilizará Aprendizaje Supervisado.

1.1.2 Estructura de una red neuronal

Una forma en la que se pueden desempeñar tareas de Aprendizaje de Maquina es mediante el uso de Redes Neuronales Artificiales o RNA por sus siglas, esta idea surge ya que lo que se quiere es imitar el aprendizaje humano para la realización de tareas, entonces hay que basar los modelos utilizados por el Aprendizaje de Maquina en la estructura fundamental del cerebro, este está formado por redes neuronales y estas a su vez están formadas por la conexión de múltiples neuronas.

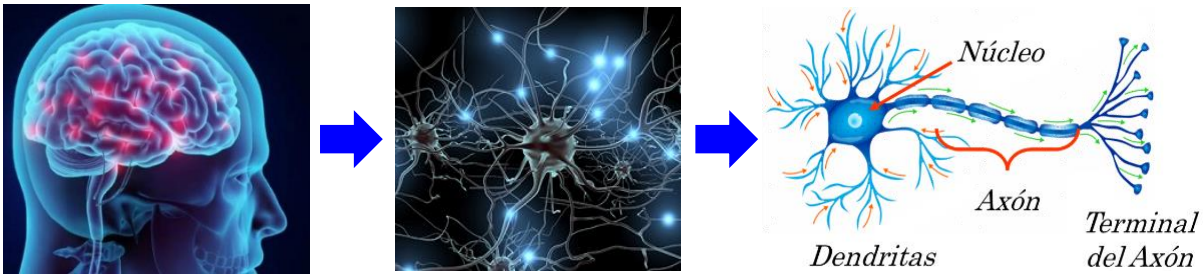


Figura 2. a) Cerebro humano, b) red neuronal biológica y c) neurona.

La función principal de una neurona es recibir impulsos eléctricos producidos por algún estimulo, si estos impulsos son lo suficientemente grandes, los transforman de alguna forma dentro del núcleo y los envían a través del axón y sus terminales a otras neuronas que harán lo mismo, por ultimo estas neuronas generarán alguna señal que desencadenará alguna respuesta, de la misma forma, se genera el llamado nodo o Neurona Artificial, una Red Neuronal Artificial está formada por diversas capas de nodos conectados unos con otros, a esta se le introduce alguna señal de entrada, se procesa y la red produce alguna señal de salida, en la Figura 3 se muestra una representación gráfica de un nodo.

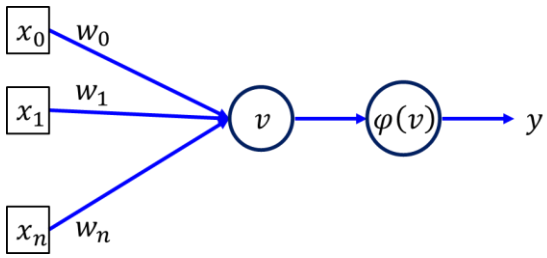


Figura 3. Estructura de un nodo simple.

La información de entrada de este nodo estará dada por el vector de características $\boldsymbol{x}$, a este se le aplica una suma ponderada v, esta puede verse como el producto matricial $\boldsymbol{xw}$ *(1.4)*, donde $\boldsymbol{w}$ es el vector de pesos, estos determinaran la importancia de la información de entrada y, por último, se aplica la función de activación $\varphi(v)$, esta determinará el comportamiento del nodo y producirá la salida de este.

La notación usada será la siguiente:

$$v^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)} = x^{(i)} w \tag{1.4}$$

Esta será la suma ponderada para el i-ésimo dato de entrenamiento, esta puede verse como el producto del i-ésimo vector de características $\boldsymbol{x^{(i)}}$ por el vector de pesos $\boldsymbol{w}$, el vector de características y el vector de pesos se pueden ver de la siguiente forma:

$$\boldsymbol{x^{(i)}} = \left(x_0^{(i)}, x_1^{(i)}, \cdots, x_n^{(i)}\right) \tag{1.5}$$

$$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} \tag{1.6}$$

$$y^{(i)} = \varphi\left(\boldsymbol{x^{(i)}}\boldsymbol{w}\right) = \varphi\left(\boldsymbol{v^{(i)}}\right) \tag{1.7}$$

La salida del nodo para el i-ésimo dato de entrenamiento será la función de activación aplicada a la suma ponderada de las entradas *(1.7)*, usualmente se le asigna a x_0 el valor de 1, de esta forma se tendrá el producto $w_0 x_0 = w_0 = b$, a este valor se le llama valor inicial o termino de sesgo ('bias term', en inglés).

Una forma sencilla de ver todos los vectores de características del conjunto de datos de entrenamiento es de la siguiente forma:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x^{(1)}} \\ \vdots \\ \boldsymbol{x^{(i)}} \\ \vdots \\ \boldsymbol{x^{(m)}} \end{pmatrix} \tag{1.8}$$

Esta es la matriz de características, donde la i-ésima fila corresponde a la entrada del i-ésimo dato de entrenamiento. De esta forma se puede calcular la

salida de todos los datos de entrenamiento, aplicando la función de activación al producto de la matriz de características por su respectivo valor de pesos, como se muestra a continuación.

$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{w} = \begin{pmatrix} \boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{x}^{(i)} \\ \vdots \\ \boldsymbol{x}^{(m)} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} v^{(1)} \\ \vdots \\ v^{(i)} \\ \vdots \\ v^{(m)} \end{pmatrix} \tag{1.9}$$

$$\boldsymbol{Y} = \varphi(\boldsymbol{V}) = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(i)} \\ \vdots \\ y^{(m)} \end{pmatrix} \tag{1.10}$$

Estas ecuaciones son para una única Neurona Artificial, esta es la red neuronal más simple que hay y muchas veces es llamada Perceptrón [1], este tipo de red se utiliza generalmente para tareas de regresión numérica y logística [1, 2, 3].

1.1.3 Regresión

Una regresión es una técnica que genera una curva a partir de una serie de datos, de tal manera de que la separación entre cada uno de estos y su proyección sobre la curva generada sea mínima.

Existen distintas técnicas para hacer una regresión polinómica, entre las que destacan el uso del método de mínimos cuadrados, el uso de la ecuación normal y el uso de un perceptrón, las dos primeras formas obtienen la regresión de forma cerrada, es decir, obtienen el resultado en un solo paso con un conjunto de fórmulas, mientras que la solución con el uso del perceptrón obtiene el resultado de forma iterativa, inicia los parámetros de la regresión con valores aleatorios y estos se va ajustando paulatinamente hasta alcanzar un valor adecuado.

MÍNIMOS CUADRADOS

El método de mínimos cuadrados es un método que produce tantas ecuaciones como el grado de la regresión que se desee realizar, como ejemplo, se verá la regresión lineal. En este caso se desea ajustar una línea recta a una serie

de datos numéricos, de tal modo que la separación entre cada uno de los puntos y su proyección sobre la recta sea mínima.

Una forma típica de representar una línea recta es de la forma.

$$y = mx + b \tag{1.11}$$

Donde m es la pendiente de la recta y b es la altura de la intersección de la recta con el eje Y como se muestra en la Figura 4. Por otra parte, la diferencia entre la altura de cada uno de los puntos y su proyección sobre la recta está dada por *(1.12)*.

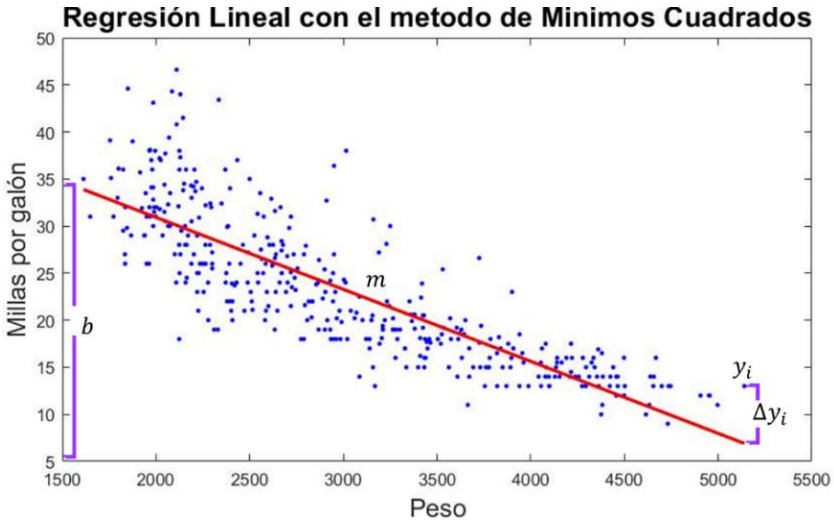


Figura 4. Regresión lineal utilizando el método de mínimos cuadrados.

$$\Delta y_i = y_i - (mx_i + b) \tag{1.12}$$

Este método calcula la suma de los cuadrados de estas diferencias como:

$$\sum_{i=1}^{n} (\Delta y_i)^2 = \sum_{i=1}^{n} [y_i - (mx_i + b)]^2 \tag{1.13}$$

Donde n es el número total de datos.

Esta suma tiene que ser lo más pequeña posible por lo que se igualan sus derivadas parciales a cero.

$$\frac{\partial}{\partial m}\left(\sum_{i=1}^{n} (\Delta y_i)^2\right) = 2\sum_{i=1}^{n} x_i[\, y_i - (mx_i + b)\,] = 0 \tag{1.14}$$

$$\frac{\partial}{\partial b}\left(\sum_{i=1}^{n}(\Delta y_i)^2\right) = 2\sum_{i=1}^{n}[\, y_i - (mx_i + b)\,] = 0 \qquad (1.15)$$

Se puede obtener a b despejando *(1.15)*.

$$b = \frac{1}{n}\sum_{i=1}^{n} y_i - m\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) = \langle y\rangle - m\langle x\rangle \qquad (1.16)$$

Donde $\langle x\rangle$ y $\langle y\rangle$ son los promedios de los valores de entrada (x_i) y sus salidas (y_i), respectivamente. Sustituyendo este valor en *(1.14)* se tiene:

$$\sum_{i=1}^{n} x_i\big[\, y_i - \big(mx_i + (\,\langle y\rangle - m\langle x\rangle\,)\big)\,\big] = 0 \qquad (1.17)$$

$$m = \frac{\sum_{i=1}^{n} x_i(\, y_i - \langle y\rangle\,)}{\sum_{i=1}^{n} x_i(\, x_i - \langle x\rangle\,)} \qquad (1.18)$$

Entonces, los valores óptimos de para una regresión lineal son:

$$b = \langle y\rangle - m\langle x\rangle \qquad (1.19)$$

$$m = \frac{\sum_{i=1}^{n} x_i(\, y_i - \langle y\rangle\,)}{\sum_{i=1}^{n} x_i(\, x_i - \langle x\rangle\,)} \qquad (1.20)$$

Esta es una forma cerrada de hacer una regresión lineal, de esta forma, la regresión se obtiene en un solo paso con el uso de dos ecuaciones, pero esta no es la forma más eficiente de obtener una regresión ya que para una regresión de primer orden se tienen dos ecuaciones, pero cuando se trata de hacer una regresión de orden n, se generan $n+1$ ecuaciones distintas, sin embargo, otra forma más directa de hacerlo es con el uso de la ecuación normal.

ECUACIÓN NORMAL.

Otra forma cerrada para encontrar la curva dada por *(1.21)* es mediante el uso de la ecuación normal de *(1.22)*. Este método se puede utilizar para realizar una regresión de grado n.

$$\hat{y} = \theta_0 + \theta_1 x + \cdots + \theta_n x^n = \boldsymbol{x\hat{\theta}} \qquad (1.21)$$

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{y} \tag{1.22}$$

Donde $\hat{y}$ es la altura de la proyección sobre la curva para algún dato de entrada, obtenido con los términos de $\widehat{\boldsymbol{\theta}}$, éste es el vector fila con los valores óptimos de los coeficientes de regresión para la curva, $\boldsymbol{y}$ es el vector con las alturas de todos los puntos y $\boldsymbol{X}$ es una matriz con las distintas potencias de los datos de entrada $\boldsymbol{x}$, como se muestra en *(1.26)*.

$$\boldsymbol{X} = \begin{pmatrix} x_1^0 & \cdots & x_1^i & \cdots & x_1^n \\ \vdots & & \vdots & & \vdots \\ x_j^0 & \cdots & x_j^i & \cdots & x_j^n \\ \vdots & & \vdots & & \vdots \\ x_m^0 & \cdots & x_m^i & \cdots & x_m^n \end{pmatrix} = \begin{pmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_j} \\ \vdots \\ \boldsymbol{x_m} \end{pmatrix} \tag{1.23}$$

$$\widehat{\boldsymbol{\theta}} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix} \tag{1.24}$$

$$\widehat{\boldsymbol{y}} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{pmatrix} \tag{1.25}$$

$$\boldsymbol{y} = \boldsymbol{X}\widehat{\boldsymbol{\theta}} \tag{1.26}$$

Donde x_j es el j-ésimo dato de entrada y está elevado a la i-ésima potencia. De esta forma se pueden calcular de forma directa los valores óptimos para la curva deseada, esta es una forma rápida de obtener una regresión, sin embargo, cuando se desea hacer una regresión de orden superior, la matriz $\boldsymbol{X}$ aumenta su tamaño y el tiempo para hacer los cálculos de *(1.22)* también crece, una forma "rápida" que se aproxima a estos resultados es hacer la regresión mediante el uso de un perceptrón, para regresiones de orden superior, este método puede resultar ser más rápido.

PERCEPTRÓN

La regresión polinómica se puede hacer de manera iterativa mediante el uso del perceptrón, en la Figura 5 se muestra un esquema de este.

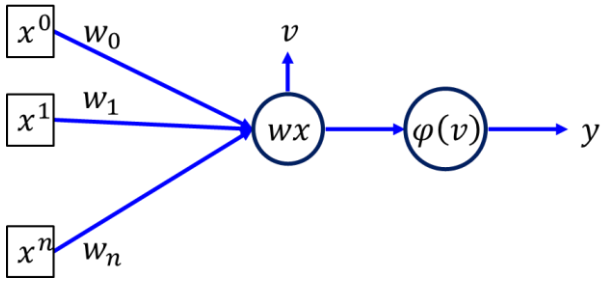


Figura 5. Esquema del perceptrón usado para generar una regresión de grado n.

El vector de características para la regresión esta dado por las distintas potencias de la variable de entrada x y $\boldsymbol{w}$ es el vector de los coeficientes (pesos) para estas potencias, como se muestra en *(1.28)* y la suma ponderada para este caso está dado por *(1.29)*.

$$\boldsymbol{x^T} = (x^0 \quad x^1 \quad \cdots \quad x^n) \tag{1.27}$$

$$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} \tag{1.28}$$

$$\hat{y} = v = \boldsymbol{x^T w} = \sum_{i=0}^{n} x^i w_i \tag{1.29}$$

Donde $\hat{y}$ es la altura de la proyección de los puntos sobre la curva y n es el grado de la regresión, en este caso, la función de activación es lineal, por lo que la salida y la suma ponderada son iguales.

Este método obtiene de forma iterativa los valores del vector $\boldsymbol{w}$, calculando el desempeño del modelo y después corrigiendo los pesos, para medir el desempeño se utiliza una función de costo, en este caso se utiliza la medida del Error Cuadrático Medio (MSE, por sus siglas en inglés), para esto se utiliza la matriz de características *(1.30)*, los valores óptimos de $\boldsymbol{w}$ serán los que minimicen esta función de costo *(1.31)*.

$$\boldsymbol{X} = \begin{pmatrix} x_1^0 & \cdots & x_1^i & \cdots & x_1^n \\ \vdots & & \vdots & & \vdots \\ x_j^0 & \cdots & x_j^i & \cdots & x_j^n \\ \vdots & & \vdots & & \vdots \\ x_m^0 & \cdots & x_m^i & \cdots & x_m^n \end{pmatrix} = \begin{pmatrix} \boldsymbol{x_1^T} \\ \vdots \\ \boldsymbol{x_j^T} \\ \vdots \\ \boldsymbol{x_m^T} \end{pmatrix} \tag{1.30}$$

$$MSE(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{x_i^T w} - y^{(i)}\right)^2 \tag{1.31}$$

Para minimizar la función de costo se utiliza el gradiente descendiente, este se puede utilizar de tres maneras, está el gradiente descendiente por lotes, el gradiente descendiente estocástico y el gradiente descendiente por mini lotes.

El descenso de gradiente es un método que encuentra los mínimos de una función de costo, para que este método encuentre el mínimo absoluto de la función y no uno local, ésta debe tener forma de cuenco.

Para implementar el gradiente descendiente a alguna función de costo, primero se calculan sus derivadas parciales con respecto a cada uno de los parámetros del vector $\boldsymbol{w}$, como se muestra en *(1.32)*.

$$\frac{d}{dw_j} MSE(\boldsymbol{x}, \boldsymbol{w}) = \frac{2}{m_i}\sum_{i=1}^{m}\left(\boldsymbol{x_i^T w} - y^{(i)}\right)x_j^{(i)} \tag{1.32}$$

Donde se calcula la parcial de la función con respecto al j-ésimo elemento del vector de pesos. Estas derivadas se pueden calcular de manera individual o calcularlas todas utilizando la formula *(1.33)*.

$$\Delta_W\, MSE(X, \boldsymbol{w}) = \begin{pmatrix} \frac{d}{dw_0} MSE(\boldsymbol{X}, \boldsymbol{w}) \\ \frac{d}{dw_1} MSE(\boldsymbol{X}, \boldsymbol{w}) \\ \vdots \\ \frac{d}{dw_n} MSE(\boldsymbol{X}, \boldsymbol{w}) \end{pmatrix} = \frac{2}{m}\boldsymbol{X^T}(\boldsymbol{Xw} - \boldsymbol{y}) \tag{1.33}$$

Este método se llama descenso de gradiente por lotes ya que calcula el gradiente utilizando todo el conjunto de datos de entrenamiento, se dirá que ha

transcurrido una época cada vez que se entrene el modelo con todos los datos de entrenamiento, una vez obtenido el resultado de *(1.34)*, se puede actualizar el vector de pesos como:

$$\boldsymbol{w}^{(k)} = \boldsymbol{w}^{(k-1)} - \eta \boldsymbol{\Delta}_{\boldsymbol{W}}\, MSE(\boldsymbol{X}, \boldsymbol{w}) \qquad (1.34)$$

Donde $\boldsymbol{w}^{(k)}$ es el valor actualizado de peso para la k-ésima época [1], este se calcula como la suma del valor de peso anterior menos el gradiente de la función de costo multiplicado por una taza de aprendizaje, ésta determinara la rapidez con la que se actualizan los pesos, si es muy baja, el modelo se actualizará lentamente y para alcanzar un resultado aceptable se necesitaran muchas épocas y, por el contrario, si es muy alta, la actualización de pesos será grande, por lo que el error puede llegar a divergir y con cada iteración aumentar, lo que llevara a malos resultados, por lo que, es necesario seleccionar adecuadamente una taza de aprendizaje para tener una actualización de pesos rápida y evitar que el modelo diverja.

El gradiente descendiente por lotes es un método de actualización lento ya que para actualizar los pesos de un nodo, tiene que haber pasado una época, por lo que se desarrolló otro modelo llamado gradiente descendiente estocástico, este, a diferencia del método por lotes, modifica los pesos del nodo con un dato de entrenamiento a la vez, es decir, selecciona un dato aleatoriamente, después calcula su medida de desempeño y con esta actualiza los valores de pesos, este método es rápido, pero es susceptible a las variaciones de los datos de entrenamiento, lo que produce fluctuaciones en los resultados, debido a esto se creó el método de gradiente descendiente por mini lotes que aprovecha la estabilidad del método por lotes y la rapidez del método estocástico, este método divide al conjunto total de datos de entrenamiento en subgrupos y con estos actualiza los pesos del nodo.

Para ver el comportamiento de un perceptrón usado para hacer una regresión, se realizará una regresión a un conjunto de datos generados artificialmente, utilizando el método de gradiente descendiente por lotes. El conjunto de datos contendrá 100 puntos generados aleatoriamente en el intervalo $(-3, 3)$ y los valores de Y de los puntos estará dado por una ecuación cuadrática más valores aleatorios que simularan el ruido en una señal, los datos para esta

regresión se muestran e la Figura 6, ya que estos datos se generaron a partir de una ecuación cuadrática, la regresión generada será de segundo orden.

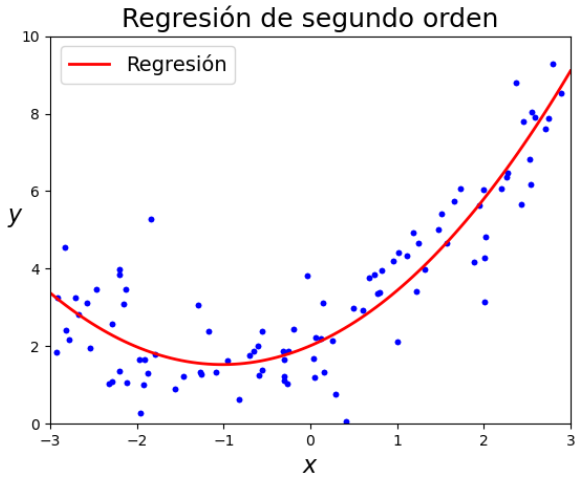


Figura 6. Regresión cuadrática aplicada a un conjunto de datos sintéticos, en esta se muestra en color azul los datos utilizados y en rojo la curva generada con la regresión.

Los parámetros utilizados para generar la curva fueron $\boldsymbol{w} = (2, 1, 0.5)$ y los obtenidos por el modelo fueron $\boldsymbol{w} = (2.0121911, 0.95469633, 0.4696792)$, como se puede ver, los valores obtenidos por el modelo son cercanos a los valores reales. En este caso, se sabe que la regresión que se debe hacer es de segundo orden ya que los datos fueron generados por una ecuación cuadrática, pero cuando se trata de analizar datos reales, no se sabe si existe una función generadora de estos, por lo que se al intentar hacer una regresión se puede tener un grado muy alto o bajo, lo que puede generar el problema de sobreajuste o subajuste.

Para entender estos conceptos, se generó una serie de datos sintéticos a partir de una ecuación cuadrática, de esta manera se obtuvo una serie de datos de "entrenamiento", usado para obtener los coeficientes de la regresión, utilizando distintos grados de regresiones para ver con cual se obtiene el mejor resultado y se usó una serie de datos de "prueba" para evaluar el desempeño de las regresiones, estos resultados se muestran en la Figura 7.

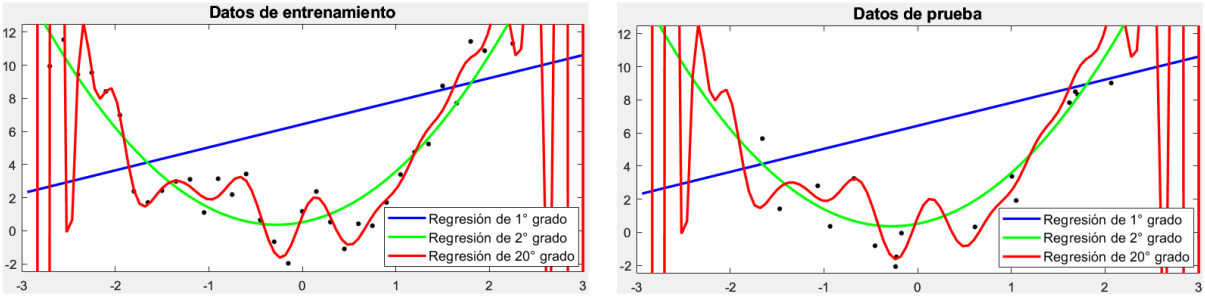


Figura 7. Regresión polinómica aplicada a una serie de datos de a) entrenamiento y b) prueba.

Como se puede ver, la regresión lineal no se ajusta correctamente a los datos de entrenamiento ya que la mayoría de los puntos se alejan mucho de la curva, este comportamiento también se observa en los datos de prueba, este es un caso típico de subajuste y se debe a que el modelo es muy simple para los datos de entrenamiento, lo que produce malos resultados.

Por otro lado, la regresión de vigésimo grado se ajusta muy bien a los datos de entrenamiento, incluso algunos puntos están sobre la curva, lo que parecería ser un buen resultado, sin embargo, para los datos de prueba esto no sucede ya que la mayoría de estos puntos se salen de la curva, este es un caso de sobreajuste, esto se debe a que el modelo es muy complejo para los datos de entrenamiento.

Por último, la regresión de segundo grado se ajusta de manera similar para los datos de entrenamiento y prueba, cuando esto sucede, se dirá que se tiene una buena generalización.

Esto era de esperarse ya que sabemos que los datos fueron generados a partir de una ecuación cuadrática, pero al analizar mediciones reales, no se sabe si existe una función generadora de los datos, lo que muchas veces lleva problemas de sobreajuste o subajuste, entonces, ¿cómo se puede saber si un modelo está sobreajustado o sub ajustado?

Una forma de saber esto es mediante las llamadas curvas de aprendizaje, pero antes de pasar a esto es necesario conocer el concepto de validación y validación cruzada.

1.1.4 Validación y validación cruzada

Una forma empírica de saber si un modelo esta sobre ajustado es mediante el proceso de validación.

El proceso de validación divide a los datos de entrenamiento en dos grupos, uno de formación y otro de validación, esto lo hace con una relación 8:2, entrena el modelo con los datos de formación y lo evalúa con los datos de validación, en este caso, en cada época, se entrena y se evalúa el modelo con los mismos grupos de formación y validación.

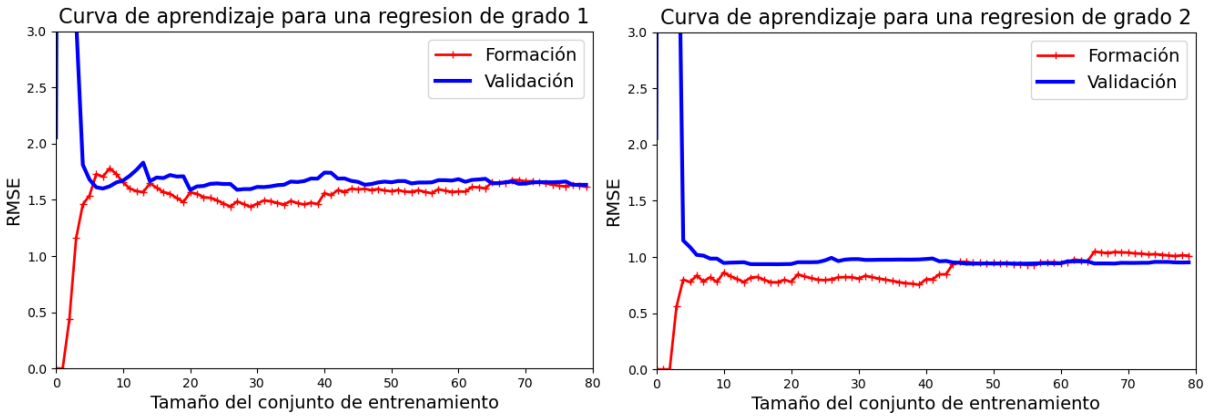
Si en este proceso el modelo tiene un buen desempeño al trabajar con los datos de formación, pero una mala generalización a los datos de validación, entonces se puede decir que el modelo estará sobre ajustado, por otra parte, si el modelo tiene un mal desempeño en ambos grupos, se puede decir que el modelo esta subajustado.

Una variación del proceso de validación es la validación cruzada, este proceso también divide a los datos de entrenamiento en estos grupos con la misma relación, pero a diferencia de la validación, los genera de forma aleatoria y en cada época se modifican estos conjuntos, lo que puede llevar a disminuir un poco el problema de sobreajuste.

1.1.5 Curvas de aprendizaje

Utilizando los procesos anteriores se puede generar la llamada curva de aprendizaje, en ésta se compara el desempeño del modelo para los datos de formación y de validación contra el número de datos del conjunto de entrenamiento.

En la Figura 8 se muestran las curvas de aprendizaje obtenidas a partir de distintas regresiones aplicadas a los datos del ejemplo anterior.



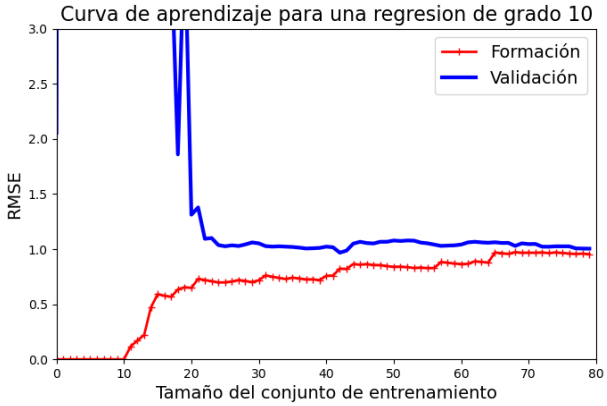


Figura 8. Curvas de aprendizaje creadas a partir de distintos modelos de regresión polinómica aplicado a un conjunto de datos sintéticos, utilizando la raíz cuadrada del error cuadrático medio (RMSE) como medida de desempeño.

Como se puede ver en las tres gráficas, la separación entre las curvas generadas con el conjunto de formación y validación empieza siendo muy grande y a medida que aumenta el número de datos de entrenamiento, esta separación disminuye, esto se debe a que, si se utilizan conjuntos de entrenamiento pequeños, se tiene muy poca información del problema que se desea analizar, lo que lleva a tener malos resultados.

En la curva de aprendizaje del modelo generado por una regresión de primer orden, las curvas generadas se acercan rápidamente con conjuntos de entrenamiento pequeños, e incluso, llegan a cruzarse cuando se tienen 5 datos y para conjuntos más grandes, esta separación vuelve a aumentar, esto significa que el modelo funciona bien cuando se tienen conjuntos de entrenamiento pequeños, pero para conjuntos más grandes, el modelo ya no funciona tan bien, lo que significa que el modelo está subajustado.

Por otra parte, en la curva de aprendizaje del modelo generado por una regresión de décimo grado, la separación entre las curvas empieza a disminuir con conjuntos de 20 datos de entrenamiento, después de esto, se acercan paulatinamente pero nunca se interceptan, esto significa que, aunque se tenga un conjunto de datos de entrenamiento muy grande, esos errores nunca serán iguales, esto nos lleva al problema de sobreajuste ya que el modelo tiene una mala generalización.

Finalmente, en la curva de aprendizaje generada por el modelo de regresión de segundo orden, a partir de 4 datos de entrenamiento esta separación

disminuye drásticamente y con forme aumenta el número de datos, esta separación disminuye gradualmente, después las curvas se interceptan cuando el conjunto de datos es lo suficientemente grande, esta separación aumenta un poco, pero disminuye nuevamente, de aquí se puede concluir que el modelo adecuado es el generado por una regresión de segundo orden.

Para realizar tareas de regresión se puede utilizar el perceptrón, pero para tareas más complejas, el perceptrón ya no funciona más y es necesario el uso de otro tipo de redes, formadas por múltiples columnas de nodos, conectados unos con otros, utilizando distintas funciones de activación, a esta se le conoce como red neuronal profunda [1, 2, 4, 5].

Todo lo que se ha visto hasta el momento, forma parte de aprendizaje de máquina, este tipo de aprendizaje funciona bien cuando las tareas que se desean analizar involucran datos simples, pero cuando se trata de realizar tareas que involucran datos complejos, este tipo de aprendizaje ya no es tan eficiente, por lo que es necesario utilizar lo que se conoce como aprendizaje profundo.

1.2 APRENDIZAJE PROFUNDO

La diferencia entre Aprendizaje de Máquina y Aprendizaje Profundo es que el Aprendizaje de Máquina solo está compuesto por una Red Neuronal como las que se han visto, esta se encarga de analizar la información de entrada sin ser modificada y producir alguna predicción, esto puede llevar a malos resultados ya que los datos analizados pueden contener señales ruidosas que degradan el entrenamiento de una red, estos resultados pueden mejorar si los datos de entrada se preprocesan manualmente, de esta manera se puede llegar a resaltar algunos detalles significativos para tener mejores predicciones.

Por otra parte, el Aprendizaje Profundo está compuesto por dos etapas, la primera es un extractor de características que se encarga de resaltar automáticamente las características importantes de los datos para que se puedan analizar más fácilmente y la segunda parte está formada por una Red Neuronal como las que se han visto, de esta manera se preprocesan los datos de una manera óptima, lo que lleva a tener mejores predicciones, por lo que el Aprendizaje Profundo es más eficiente para analizar datos complejos.

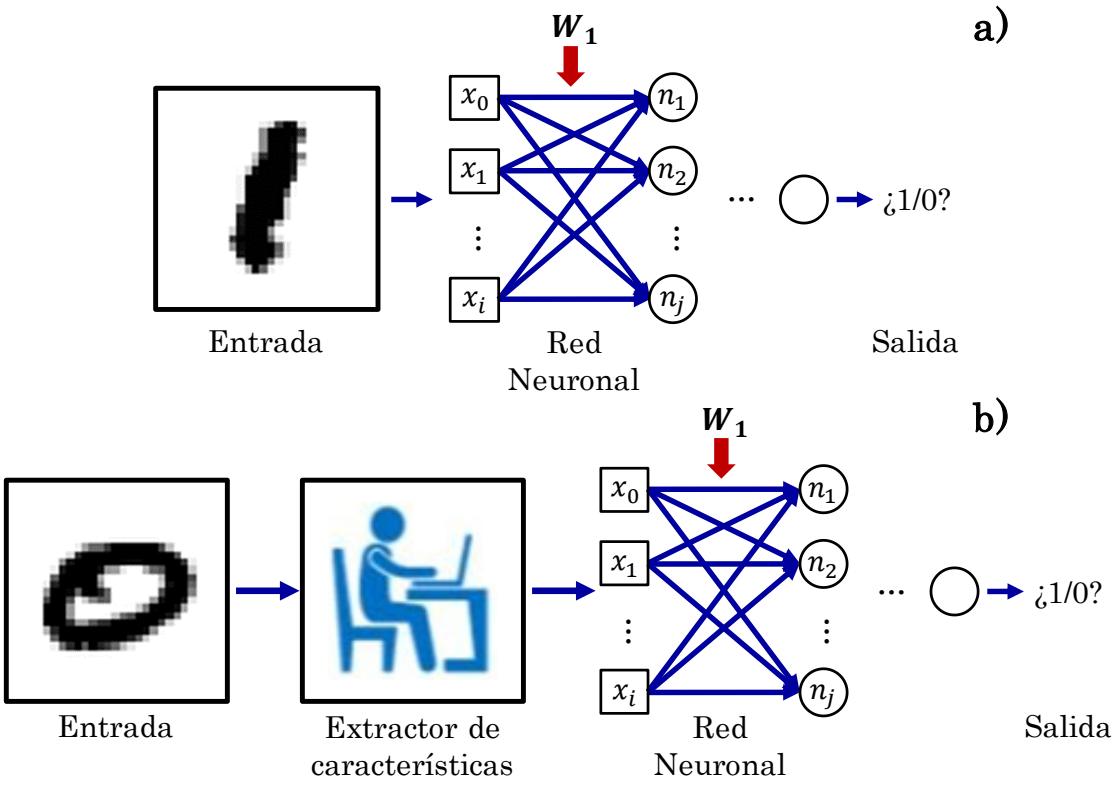


Figura 9. Esquema de a) Aprendizaje de Máquina y b) Aprendizaje Profundo

En este trabajo se analizarán imágenes del entorno del vehículo, por lo que se ocupará Aprendizaje Profundo con una red de convolución como extractor de características y una red de regresión para hacer las predicciones.

1.2.1 Red neuronal convolucional

Para el reconocimiento de patrones en imágenes no se procesa directamente las imágenes originales, primero se tienen que resaltar algunos detalles significativos, es por esto por lo que se usa una red neuronal de convolución que filtra las imágenes originales y de esta forma extrae características importantes de ellas [1, 2, 5].

Una red de convolución está formada por múltiples capas de convolución y reducción, la capa de convolución puede verse como un grupo de filtros digitales que extraen la información relevante de la imagen original y la descompone en imágenes más fáciles de analizar, después de esto, se utiliza una función de activación que resalta la información relevante de estas y, para reducir su tiempo de procesamiento, se utiliza una capa de reducción que concentra la información de una vecindad de pixeles a un solo pixel. Por último, esta información se junta y

se reordena como un vector columna, esta será la salida de la red de convolución [1, 2, 5].

La red convolucional realiza la convolución bidimensional entre la imagen original y los filtros digitales [1, 2, 3, 4] y obtiene como resultado una serie de imágenes (matrices) más pequeñas en las que resaltan algunas características importantes, después se aplica una función de activación a las imágenes filtradas, esta función resalta aún más estas características, posteriormente, estas matrices pasan por un filtro de reducción que promedia la información de una vecindad de pixeles y los reduce a un solo pixel, estas vecindades suelen ser de tamaño 2x2 o 3x3. Finalmente, esta información es reorganizada como un solo vector columna, este será la salida de la red de convolución.

Una red de convolución es una técnica de extracción de características, generalmente después de aplicar este tipo de redes se ocupan otras como las de clasificación, regresión o regresión logística [1], tomando como entrada la salida de la red de convolución, en la Figura 10 se muestra la estructura de una red neuronal formada por una red de convolución y una red de regresión.

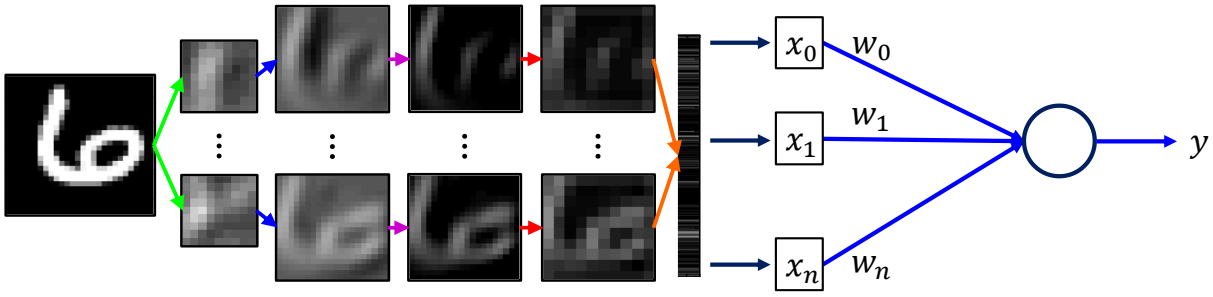


Figura 10. Estructura de una red neuronal formada por una red de convolución como extractor de características y una red de regresión para la toma de decisiones.

Una red de convolución es un método muy eficiente de extracción de características y funciona muy bien para tareas en las que el tiempo no es tan relevante, sin embargo, para tareas en que el tiempo de procesamiento es de vital importancia (procesamiento en tiempo real), este método ya no es tan eficiente ya que éste puede llegar a ser grande, por lo que se pueden utilizar otro tipo de técnicas de extracción de características y análisis de datos como lo es el método de Análisis de Componentes Principales (PCA), la Regresión de Componentes Principales (PCR) o la Proyección a Estructuras Latentes (PLS).

1.3 ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

Como se vio en la sección anterior, una forma de reducir la carga computacional y el tiempo de ejecución de una red de convolución es reduciendo la resolución de las imágenes analizadas, promediando la información de una vecindad de pixeles, sin embargo, si se reduce la información de esta forma, se pueden llegar a perder detalles significativos de los datos analizados, lo que degrada la calidad del entrenamiento de una red.

Una forma de reducir los datos sin tanta pérdida de información es mediante el uso de la reducción de dimensionalidad, un método común para hacer esto es el Análisis de Componentes Principales (o PCA, por sus siglas en inglés), este método toma una serie de datos de entrada, representado en sus dimensiones originales y los reescribe en términos de otras dimensiones llamadas componentes principales, donde el número de componentes principales que el mismo número de dimensión del conjunto original y los primeros componentes principales contienen la mayor cantidad de información.

Utilizando esta técnica, se pueden expresar una serie de datos utilizando solo las primeras componentes principales y así se puede simplificar un problema ya que se puede pasar a varios cientos o miles de dimensiones pue no se pueden visualizar fácilmente a dos o tres que ya se pueden visualizar y de esta manera se puede encontrar alguna relación o patrón entre los datos que se desee analizar.

Este método se puede utilizar para el análisis de imágenes, en este caso, se puede pasar de una dimensión rxc, donde r es el número de renglones y c de columnas de la imagen, en este caso, la reducción de dimensionalidad se podría pensar como encontrar los "ingredientes fundamentales" que conforman a las imágenes analizadas, si se analizaran las imágenes para alguna tarea en específico, por ejemplo, para el análisis de las fotos de dígitos escritos a mano, se podría descubrir que las imágenes contienen ciertas similitudes que se pueden cuantizar y de esta manera se puede describir.

Un método para encontrar todos los componentes principales es mediante la Descomposición de Valor Singular o SVD (Singular Value Decomposition, en

inglés) que descompone la matriz de datos de entrada $\boldsymbol{X}$ como el producto de tres matrices (1.35).

$$\boldsymbol{X} = \boldsymbol{U\Sigma V^T} \tag{1.35}$$

Donde X es una matriz de datos de tamaño mxn, $\boldsymbol{V}$ es una matriz cuadrada ortogonal de tamaño nxn que contiene todos los componentes principales de X, U es una matriz cuadrada de dimensiones mxm y $\boldsymbol{\Sigma}$ es una matriz "diagonal" de tamaño mxn, estas matrices se ven de la siguiente forma:

$$U = \begin{bmatrix} u_{1,1} & \cdots & u_{1,m} \\ \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,m} \end{bmatrix} \tag{1.36}$$

$$\boldsymbol{V} = \begin{bmatrix} v_{1,1} & \cdots & v_{1,n} \\ \vdots & \ddots & \vdots \\ v_{n,1} & \cdots & v_{n,n} \end{bmatrix} \tag{1.37}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n \\ \bar{0} & \cdots & \bar{0} \end{bmatrix} si\ n < m \tag{1.38}$$

$$\boldsymbol{\Sigma} = \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \sigma_m & 0 \end{array}\right] si\ m < n \tag{1.39}$$

Donde $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$ son los valores singulares de X y $p = min(m, n)$.

Para análisis de imágenes, generalmente se utiliza SVD reducido, para esto solo se toma la parte cuadrada de la matriz $\boldsymbol{\Sigma}$, eliminando las columnas o filas que son cero de la matriz original, como se muestra en *(1.40)*.

$$\boldsymbol{\Sigma_1} = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{bmatrix} donde\ p = min(m, n) \tag{1.40}$$

Para esto, las matrices $\boldsymbol{U}$ y $\boldsymbol{V}$ deben tener las siguientes dimensiones:

$$\boldsymbol{U_1} \rightarrow [m, p], \qquad \boldsymbol{\Sigma_1} \rightarrow [p, p], \qquad \boldsymbol{V_1} \rightarrow [p, n] \tag{1.41}$$

Donde $\boldsymbol{U}_1$ tiene las primeras p columnas de $\boldsymbol{U}$ y $\boldsymbol{V_1}$ tiene las primeras p filas de $\boldsymbol{V}$, entonces la matriz X puede escribirse como:

$$\boldsymbol{X} = \boldsymbol{U_1 \Sigma_1 V_1} \qquad (1.42)$$

En este caso $\boldsymbol{X}$ es la matriz de características que contiene toda la información de las imágenes que se desean analizar, donde m es el número total de pixeles de las imágenes y n es el número total de imágenes que se desean analizar.

Una vez que se obtienen todos los componentes, se puede realizar una regresión con un conjunto de éstos, este método se puede ocupar para estimar una predicción de la salida de algún dato de entrada, a esta técnica se le conoce como Regresión de Componentes Principales.

1.3.1 Regresión de Componentes Principales (PCR)

El método de Regresión de Componentes Principales realiza dos procesos, primero calcula el PCA de la entrada de una serie de datos de entrenamiento y después calcula una regresión utilizando sus salidas, de esta forma se le asigna un valor de peso a los componentes principales que corresponde a su relevancia para obtener las predicciones de los datos de entrenamiento.

La desventaja de esta técnica es que el método de PCR es una técnica "no supervisada" ya que calcula las componentes principales basándose en la información de los datos de entrada, sin tomar en cuenta los datos de salida, por lo que, si se hace la regresión con estos, se puede tener información poco relevante para los datos de salida, lo que puede degradar el desempeño de esta técnica. Debido a esto, se utilizará la técnica de PLS que es similar a PCR, pero, a diferencia de ésta, este es un método "supervisado" por lo que puede generar mejores predicciones.

1.4 PROYECCIÓN A ESTRUCTURAS LATENTES (PLS)

El método de Proyección a Estructuras Latentes o Regresión de Mínimos Cuadrados Parciales es una técnica de reducción de dimensionalidad similar al método de Análisis de Componentes Principales, pero a diferencia de ésta, el método de PLS calcula los componentes con respecto los datos de entrada y salida.

Este método encuentra un conjunto de variables llamadas puntuaciones de X (T), estas son combinaciones lineales de las variables de entrada $\boldsymbol{X}$ por alguna matriz de pesos $\boldsymbol{W}$, estas puntuaciones son estimaciones de las variables latentes

de $\boldsymbol{X}$, entonces se puede reescribir $\boldsymbol{X}$ como el producto de sus puntuaciones por una matriz de cargas más una matriz de residuos que generalmente es pequeña.

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W}^{*} \tag{1.43}$$

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^{T} + \boldsymbol{E} \tag{1.44}$$

De la misma forma, se puede expresar a la salida Y como el producto de las puntuaciones de $\boldsymbol{Y}$ por una matriz de cargas más una matriz de residuos que también es pequeña.

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{C}' + \boldsymbol{G} \tag{1.45}$$

Una de las propiedades de la matriz de puntuaciones de $\boldsymbol{X}$ es que son buenos predictores de $\boldsymbol{Y}$, entonces $\boldsymbol{Y}$ se puede escribir como *(1.46)*, donde $\boldsymbol{F}$ es la diferencia entre la salida predicha con las puntuaciones de $\boldsymbol{X}$ y la salida deseada.

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{C}^{T} + \boldsymbol{F} \tag{1.46}$$

$$\boldsymbol{F} = \boldsymbol{Y} - \boldsymbol{Y}_{\boldsymbol{Pred}} \tag{1.47}$$

$$\boldsymbol{Y}_{\boldsymbol{Pred}} = \boldsymbol{T}\boldsymbol{C}' \tag{1.48}$$

Si en esta predicción se sustituye la ecuación *(1.43)*, se tiene el resultado *(1.49)*, este se puede expresar como un modelo de regresión, ocupando solo las primeras p columnas de la matriz $\boldsymbol{W}$ y $\boldsymbol{C}$, como se muestra en *(1.50)* y *(1.51)*, estas originalmente tienen n columnas.

$$\boldsymbol{Y}_{\boldsymbol{Pred}} = \boldsymbol{X}\boldsymbol{W}^{*}\boldsymbol{C}' \tag{1.49}$$

$$\boldsymbol{W}^{*} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix} \rightarrow \boldsymbol{W}^{*} = \begin{bmatrix} w_{11} & \cdots & w_{1p} \\ \vdots & & \vdots \\ w_{n1} & \cdots & w_{np} \end{bmatrix} \tag{1.50}$$

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & & \vdots \\ c_{q1} & \cdots & c_{qn} \end{bmatrix} \rightarrow \boldsymbol{C} = \begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & & \vdots \\ c_{q1} & \cdots & c_{qp} \end{bmatrix} \tag{1.51}$$

Donde:

$n \rightarrow$ Número de Pixeles (dimensiones)

$m \rightarrow$ Número de datos

$p \rightarrow$ Numero de componentes para la regresión

$q \rightarrow$ Número de variables de salida

El resultado de *(1.49)* se puede reescribir como *(1.52)*, donde $\boldsymbol{B}$ es la matriz de coeficientes de regresión de PLS-R, entonces la salida predicha con las primeras componentes de $\boldsymbol{X}$ se puede escribir como:

$$\boldsymbol{Y_{Pred}} = \boldsymbol{XB} \qquad (1.52)$$

El resultado de esta predicción dependerá del número de componentes usados para hacer las PLS-R, entre más componentes se utilicen se tendrá un resultado más cercano al deseado, pero esto puede llevar al problema de sobreajuste.

Una forma para generar estas matrices es mediante el uso del algoritmo NIPALS (Non-linear Iterative Partial Least Squares), que utiliza las matrices de entrada y salida $(\boldsymbol{X}, \boldsymbol{Y})$ de forma normalizada y centrada, donde $\boldsymbol{Y}$ puede tener más de una característica de salida.

El algoritmo NIPALS para una sola variable de salida se puede describir de la siguiente forma:

1. Inicialización los valores de las primeras matrices de entradas, salidas y el valor de i que en este caso, corresponde al primer componente principal:

$$\boldsymbol{X_1} = \boldsymbol{X}, \ \boldsymbol{Y_1} = \boldsymbol{Y} \quad y \quad i = 1 \qquad (1.53)$$

2. Inicialización el valor de la i-esima columna (información del i-esimo componente) de la matriz $\boldsymbol{U}$.

$$\boldsymbol{u}_i = \boldsymbol{Y_i} \qquad (1.54)$$

3. Cálculo de la i-esima columna de la matriz de pesos y normalización de la misma.

$$\boldsymbol{w}_i = \boldsymbol{X}_i' \frac{\boldsymbol{u_i}}{\boldsymbol{u}_i' \boldsymbol{u}_i} \ \rightarrow \ \boldsymbol{w}_i = \frac{\boldsymbol{w}_i}{\|\boldsymbol{w}_i\|} \qquad (1.55)$$

4. Cálculo de las puntuaciones de X:

$$\boldsymbol{t}_i = \boldsymbol{X_i w_i} \qquad (1.56)$$

5. Cálculo de los pesos de Y como:

$$\boldsymbol{c_i} = \boldsymbol{Y'_i}\frac{\boldsymbol{t_i}}{\boldsymbol{t'_i t_i}} \tag{1.57}$$

6. Actualización del valor de U como:

$$\boldsymbol{u_i} = \boldsymbol{Y'_i}\frac{\boldsymbol{c_i}}{\boldsymbol{c'_i c_i}} \tag{1.58}$$

7. Estimación de la convergencia de T de la j-esima iteración:

$$\frac{\|\boldsymbol{t_{j-1}} - \boldsymbol{t_j}\|}{\|\boldsymbol{t_j}\|} < \boldsymbol{\varepsilon} \tag{1.59}$$

   Esta debe ser "pequeña" (de un orden de 10^{-6} o 10^{-8}), si no es asi, regresa al paso 3, pero si es "pequeña", pase alpaso 8.
8. Actualización del valor de X_{i+1} y Y_{i+1}, para el calculo del siguiente componente, restando los resultados obtenidos con el componente actual.

$$\boldsymbol{p_i} = \boldsymbol{X'_i}\frac{\boldsymbol{t_i}}{\boldsymbol{t'_i t_i}} \tag{1.60}$$

$$\boldsymbol{X_{i+1}} = \boldsymbol{X_i} - \boldsymbol{t_i p_i} \tag{1.61}$$

$$\boldsymbol{Y_{i+1}} = \boldsymbol{Y_i} - \boldsymbol{t_i c_i} \tag{1.62}$$

9. Repetir los pasos 2-8 para los siguientes componentes, para $i = i + 1$, hasta llegar a el número de componentes deseado (q).

De esta fora se calcula la informacion de las distintas matrices para los distintos componentes, una vez teniendo las matrices necesarias, se calcula la matriz $\boldsymbol{B}$ que se usara para generar las predicciones para nuevos datos.

Para la generacion de datos de entrenamiento para los distintos modelos se utilizará una computadora de tarjeta simple, estas se mostrarán en el siguiente capitulo, tambien se mostrará el arreglo experimental usado para generar el prototipo del vehiculo y los programas usados para su manejo.

CAPÍTULO 2. DESARROLLO EXPERIMENTAL

2.1 COMPUTADORA DE PLACA ÚNICA

Para obtener los datos de entrenamiento se usará una computadora de placa única (SBC, por sus siglas en inglés), esta es una computadora completa que contiene todos los componentes de una computadora tradicional, soldados a una sola placa. Este tipo de computadoras son muy útiles para trabajos de robótica debido a su tamaño y a su bajo consumo energético. Los modelos propuestos para el desarrollo del prototipo son la Raspberry pi 4, la Jetson Nano y un mini pc MELE.

2.1.1 Raspberry Pi 4

La Raspberry Pi es una SBC con procesador tipo ARM de 64 bits. El sistema operativo que usa es Raspbian, este está basado en Debian y fue creado específicamente para esta tarjeta. También Existen otros sistemas que se le podrían instalar, entre los cuales están Ubuntu Core, Kali y una versión de Windows 10 para procesador tipo ARM.

Esta tarjeta cuenta con 40 conectores de entrada/Salida de propósito general o GPIO por sus siglas en inglés, estos se pueden usar para interactuar con distintos actuadores o para generar distintos protocolos de comunicación y, por otra parte, tiene conexión a WiFi y a Bluethoot por lo que se puede usar de manera remota.

Para controlar los conectores GPIO existe una librería de Python llamada GPIOZero que viene instalada por defecto y se puede usar para generar señales de entrada o salida.

Para trabajos de robótica se deben usar distintos sensores y actuadores que se pueden conectar directamente a los conectores GPIO con distintas tarjetas o de manera más compacta se puede usar el llamado Robot Hat que ya contiene algunas tarjetas.

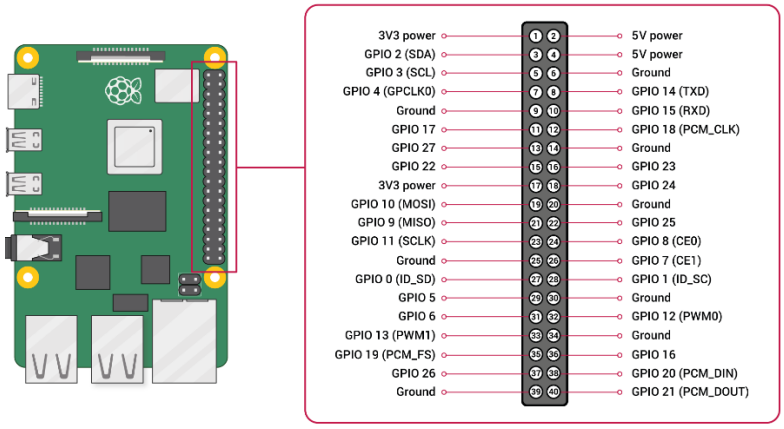


Figura 11. Conectores de propósito general GPIO de la Raspberry Pi 4.

2.1.2 Jetson nano

La Jetson Nano También es una computadora de placa única con procesador tipo ARM de 64 bits que, a diferencia de la Raspberry Pi, no cuenta con tarjeta de WiFi ni de Bluethoot, por lo que, si se desea usar de manera remota, se le tendrían que conectar estos módulos de manera externa. Esta computadora cuenta con una tarjeta gráfica que es útil para trabajos de inteligencia artificial. A esta tarjeta también se le pueden instalar los sistemas operativos ya mencionados, en la Figura 12 se muestra la Jetson Nano y sus conectores GPIO.

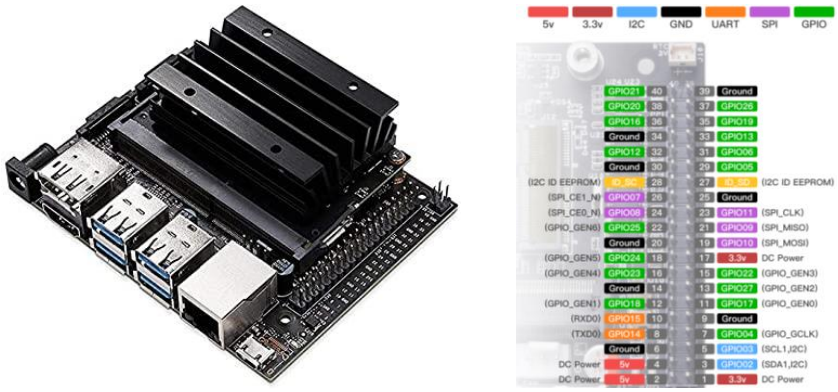


Figura 12. a) Jetson Nano, b) conectores de propósito general GPIO.

Se puede observar que estos conectores son los mismos que los de la Raspberry Pi, por lo que se puede usar un Robot Hat.

2.1.3 Mini PC MELE

A diferencia de las tarjetas anteriores, la mini PC MELE es una computadora con procesador tipo CISC, este es como el de una computadora

"convencional", tiene un Celeron de 2.7 GHz, 8 Gb de RAM y 128 de disco duro en estado sólido, expandible a 1 terabyte, también cuenta con conexión a Bluethoot y a WiFi, por lo que también se puede usar de manera inalámbrica. A esta tarjeta se le pueden instalar los sistemas operativos y programas de una computadora convencional.

Figura 13. Jetson Nano.

Existen dos arquitecturas para procesadores, está la arquitectura CISC que se especializa en instrucciones complejas y la arquitectura tipo RISC que hace operaciones reducidas, La arquitectura CISC es usada para computadoras y laptops convencionales ya que son potentes, lo malo de esta arquitectura es que consumen mucha energía a cambio de un buen rendimiento, los procesadores más comunes con este tipo de arquitectura son Core y Ryzen.

Los procesadores tipo RISC consumen poca energía a coste de rendimiento, la arquitectura ARM es un tipo avanzado de RISC, Los procesadores tipo ARM son usados comúnmente para dispositivos que usan pilas como teléfonos y tabletas, debido a que son pequeños y consumen poca energía, el procesador Exynos y Snapdragon son algunos de estos procesadores.

Los procesadores ARM se han vuelto tan eficientes que ya se están empezando a usar para laptops, debido a su bajo consumo energético, sin embargo, algunos programas para procesadores tipo RISC y ARM son incompatibles.

Para el desarrollo de los primeros prototipos se usará la Raspberry Pi y a medida que el modelo requiera más recursos, se irá escalando a distintas tarjetas.

Estos prototipos deben de ser capaces de manejarse de manera manual y autónoma, de manera manual deben de capturar imágenes de su entorno y guardarlas junto con la velocidad del vehículo, de esta manera se generarán los

datos de entrenamiento. De manera autónoma, debe ser capaz de correr una red neuronal de convolución para analizar nuevos datos y generar sus propias respuestas.

A lo largo de este trabajo se desarrollaron distintos prototipos y programas para el manejo de este, estos se modificaron para obtener un modelo más eficiente, a continuación, se mostrarán estos prototipos.

2.2 ARREGLO EXPERIMENTAL

Para el diseño del prototipo serán necesarios un par de motores que se encargarán de darle movimiento, una cámara que tomará imágenes de entorno del vehículo y un servomotor que moverá la cámara para tomar imágenes desde distintos ángulos, y una Raspberry Pi 4 para la toma de los datos de entrenamiento y por último correr la red neuronal y generar la respuesta deseada. A lo largo de este trabajo se desarrollaron distintos prototipos con el fin de mejorarlo.

2.2.1 Primer Prototipo

El primer prototipo usado fue un kit para armar que contiene una Raspberry pi 4, un Robot Hat y todas las tarjetas necesarias para controlar los distintos actuadores y entradas para los distintos sensores del Robot Hat. Se decidió utilizar este kit ya que era más barato que comprar los componentes por separado y se evitaba el tiempo de diseñar el prototipo, este kit cuenta con dos motores izquierdos y dos derechos que están conectados en paralelo y además contiene un servomotor para controlar el movimiento de la cámara y generar los datos de entrenamiento a distintos ángulos, en la Figura 14 se muestra el prototipo utilizado.

Figura 14. Primer prototipo de KITT.

Una vez teniendo el prototipo, se procedió a caracterizar los distintos actuadores, como se puede ver en la Figura 15, el servomotor tiene un comportamiento lineal respecto del ancho de pulso, por otra parte, el comportamiento de un par de llantas delantera y trasera se muestra en la Figura 16, de aquí se puede ver que el movimiento de las llantas no tiene un comportamiento lineal con respecto al voltaje entregado.

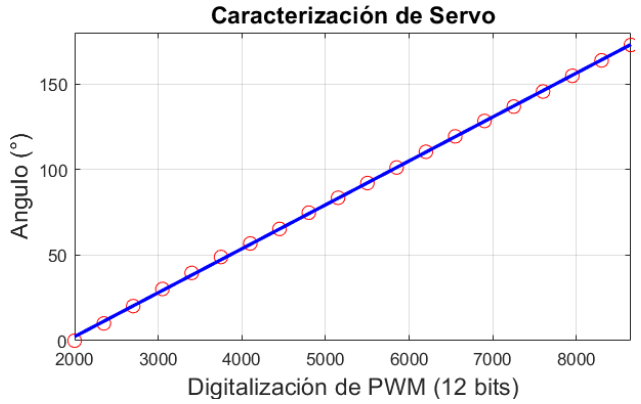


Figura 15. Caracterización del servomotor que mueve a la cámara del prototipo, en este podemos ver un comportamiento lineal entre el ancho del pulso y el ángulo, con un ajuste lineal dado por la ecuación: $\theta = 0.0257 * PWM - 48.9819$.

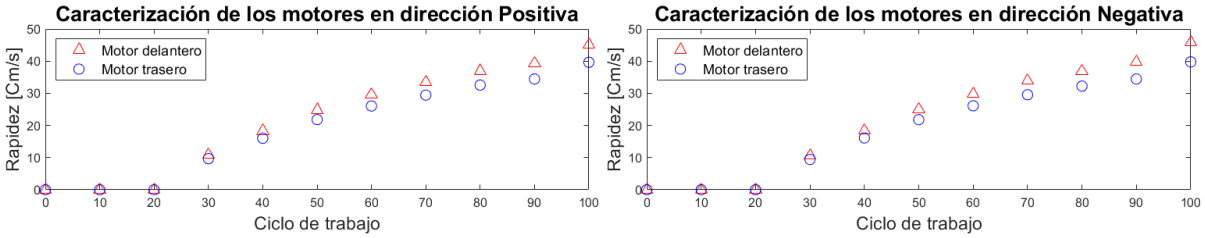


Figura 16. Caracterización de las llantas conectadas en paralelo.

Como se puede ver, las llantas traseras avanzan más lento que las delanteras, esto producirá que los motores delanteros se fuercen para arrastrar a las llantas traseras, además de que, a la hora de dar vuelta, las dos llantas traseras patinaran, consumiendo energía de más, por lo que ésta no es la mejor idea de prototipo, posteriormente se cambió a un modelo más "sencillo".

2.2.2 Segundo Prototipo

El segundo prototipo se muestra en la Figura 17, este contaba con dos llantas delanteras y una llanta trasera con movimiento libre, para modificar la dirección del vehículo se deben mover las llantas delanteras con distintas

velocidades, este segundo prototipo también cuenta con una Raspberry y un Robot Hat, este prototipo parece más sencillo que el anterior, pero tiene un mejor comportamiento.

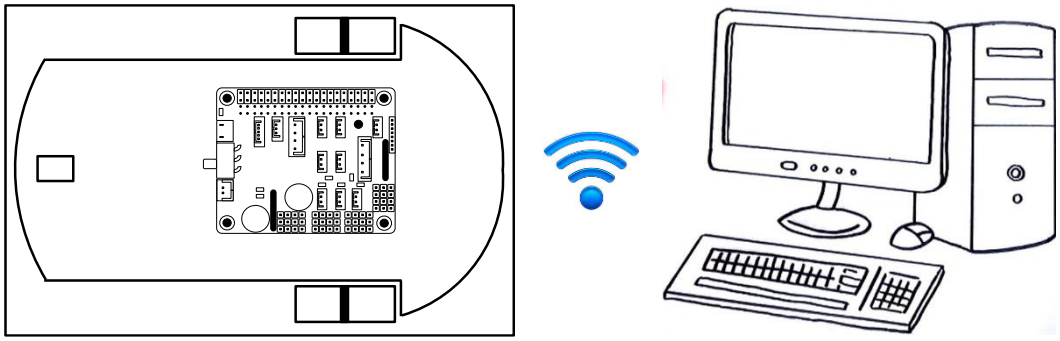


Figura 17. Segundo prototipo de KITT, este cuenta con la Raspberry Pi 4, un Robot Hat y un par de motores, para su manejo se hizo una conexión remota desde otra computadora y se utilizó el teclado como control.

Para el manejo del vehículo se hizo una conexión remota desde otra computadora y se desarrolló un programa en Python que detecta la pulsación de las flechas del teclado, así se modificó se rapidez y dirección, de esta manera es como se condujo el prototipo.

Teniendo este prototipo, se procedió con la fase de pruebas, pero se descubrió que manejarlo con el teclado no era lo más eficiente ya que resultaba complicado desplazarse, por lo que se decidió utilizar los joysticks de un control de PlayStation 4 para modificar la velocidad del vehículo, una vez teniendo el prototipo listo, se caracterizaron los motores, para esto se pusieron marcas en las llantas del prototipo y se desarrolló un programa en Python que modificaba paulatinamente el ciclo de trabajo de los motores y se grabó el comportamiento de las llantas a medida que se modificaba el ciclo de trabajo, después se guardaron estos videos y los valores del ciclo de trabajo y se analizaron en un programa en Matlab, este lo que hacía era fijarse en un solo pixel de la llanta y graficaba la intensidad del pixel a lo largo del tiempo, después se aplicó la transformada de Fourier a esta señal y se obtuvo la frecuencia de giro de las llantas, los resultados de este análisis se muestran en la Figura 18.

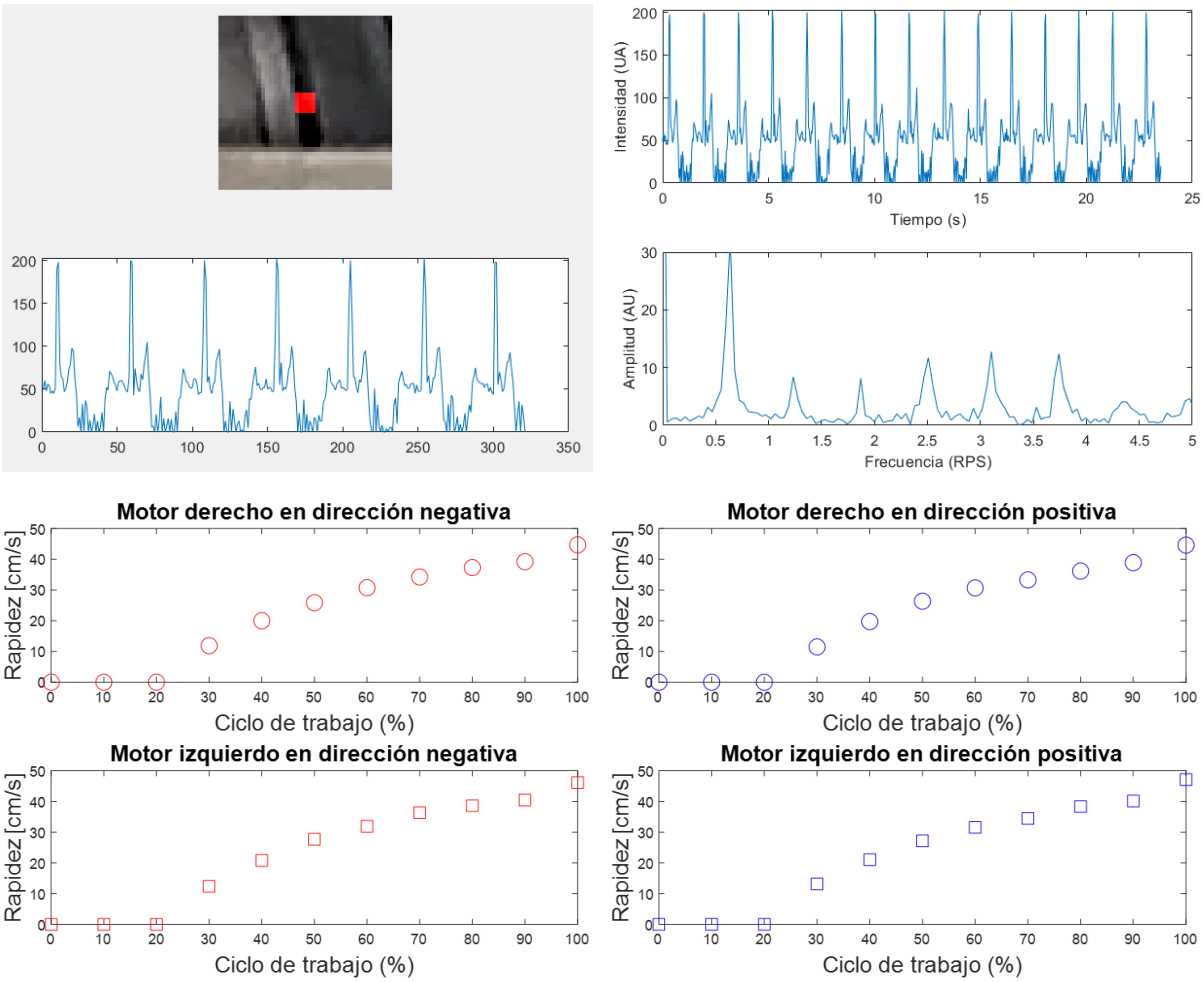


Figura 18. Caracterización de los motores en dirección positiva y negativa, la velocidad máxima alcanzada por el motor derecho e izquierdo son 47.11 y 46.13 cm/s respectivamente, por lo que, para que ambos motores se muevan a la par es necesario el factor de corrección de 0.9792 para el motor izquierdo.

Después de caracterizar los motores se creó un segundo programa, este es capaz de modificar la velocidad de las llantas del vehículo dependiendo de la posición del joystick, este programa es capaz de tomar imágenes del entorno y guardarlas junto con la información de las llantas. Este prototipo parecía ser el adecuado ya que contaba con pocos elementos y se podía conducir de manera precisa, sin embargo, cuando se empezaban a grabar los datos de entrenamiento, surgía un retraso entre la señal del control y el movimiento del vehículo, este problema se agravaba al aumentar el conjunto de datos de entrenamiento. Este problema se intentó resolver reduciendo el tamaño de las imágenes tomadas y reduciendo el número de fotos por segundo, al hacer esto se redujo un poco este

problema, pero cuando se alcanzaban los 200 datos de entrenamiento, este problema se hacía evidente y era muy difícil conducir el prototipo.

Posteriormente se descubrió el causante de este problema, esto se debía a que la Raspberry era la que, hacia todos los procesos, se encargaba de leer la información del control, mover los motores y capturar los datos de entrenamiento y como sabemos, este tipo de SBC cuenta con un procesador tipo ARM por lo que es incapaz de llevar a cabo todas estas tareas a la vez, por lo que se decidió dividir las tareas utilizando una segunda tarjeta y desarrollando un tercer prototipo más robusto.

2.2.3 Tercer Prototipo

Se desarrolló un tercer prototipo que cuenta con una placa de Arduino conectada de forma serial a la Raspberry Pi, de esta manera se dividen las tareas, el Arduino es quien recibe la señal del control y la manda a las llantas para el movimiento del vehículo y, por último, mandará esta información a la Raspberry para la toma de datos de entrenamiento, ésta solo se encargará de recibir la información, capturar imágenes y guardarlas para generar los datos de entrenamiento. Lo malo de este prototipo es que ya no se podía usar el control del PlayStation, por lo que se decidió hacer un control propio, este solo debía contener un joystick, una placa de Arduino y alguna tarjeta para la transmisión de datos al vehículo, en la Figura 19 se muestra el arreglo experimental de este prototipo.

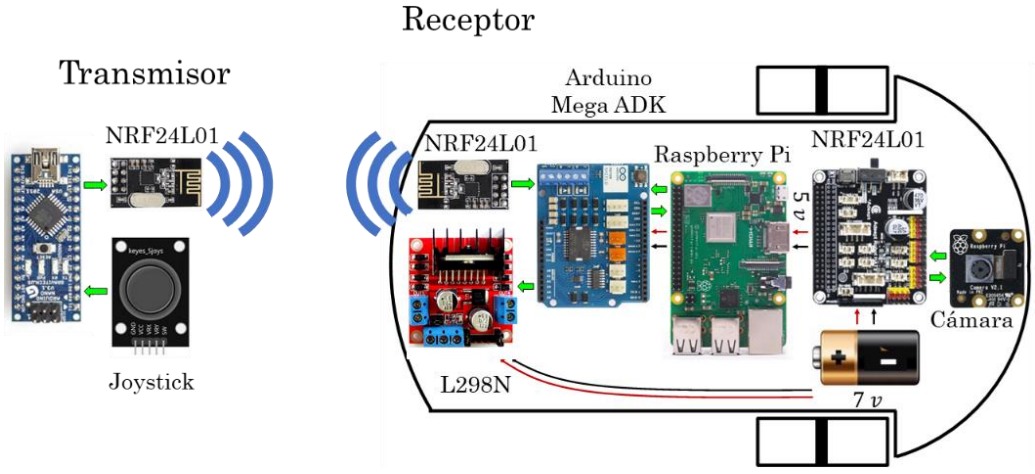


Figura 19. Arreglo experimental del tercer prototipo de vehículo.

El arreglo experimental usado para este prototipo es el siguiente: se utilizó un par transmisor/receptor con el módulo NRF24L01 para establecer la

comunicación entre estos, el transmisor está formado por un joystick, una pila de 9 volts y un Arduino NANO que es el que leerá la información del joystick y la enviará al receptor por la placa de radiofrecuencia.

Por otra parte, el receptor está formado por una pila de 7 volts, la Raspberry Pi, un Robot Hat, que se encarga de reducir voltaje de la pila y alimentar a la Raspberry, un Arduino UNO, una placa de radiofrecuencia que recibirá la información del control y un módulo L298N que se encargará de mover los motores, este se alimentará directamente de la pila. También se utilizó un par de servomotores para mover la cámara y tomar los datos de entrenamiento desde distintos ángulos, uno mueve a la cámara sobre el eje vertical y el otro sobre el horizontal.

Este sistema presentaba algunas complicaciones a la hora de conducirlo ya que, a la hora de transportarlo, surgían algunos falsos contactos ya que todos los componentes estaban conectados por medio de jumpers, esto producía que el sistema tuviera comportamientos extraños, debido a esto se decidió sustituir las conexiones del L298N y utilizar un Moto Shield que ya lo tiene incluido, esta placa se monta sobre el Arduino y ya no necesita más cables. Lo malo de usar el Motor Shield es que utiliza un par de pines que son necesarios para la comunicación con la placa de radiofrecuencia. Debido a esto se utilizó un Arduino Mega ADK que contiene más pines que pueden ser utilizados para la comunicación con la placa de radiofrecuencia.

Por último, el prototipo para el receptor quedo de la siguiente forma:

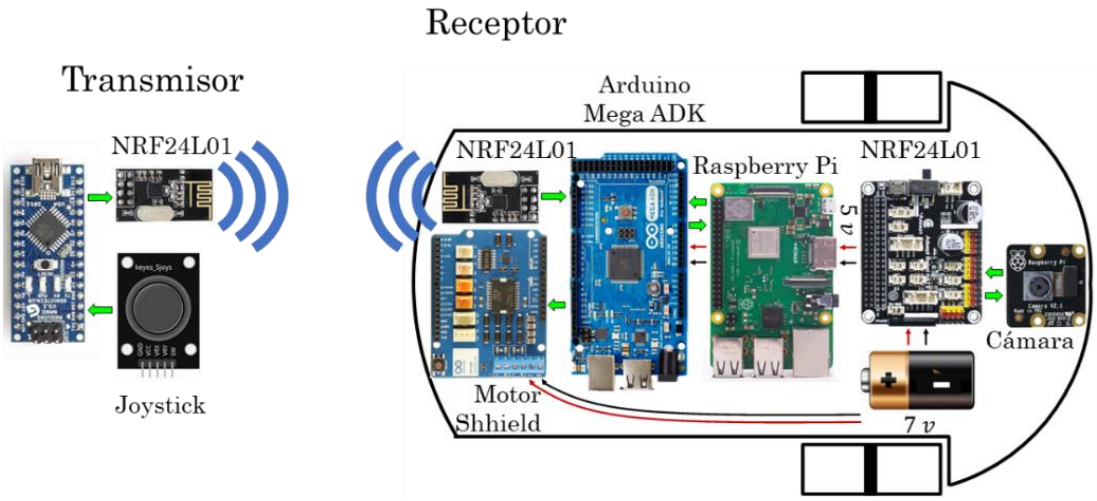


Figura 20. Arreglo experimental del tercer prototipo.

Por otra parte, también se decidió eliminar las conexiones del transmisor para evitar los falsos contactos, para esto se diseñó una placa en Proteus que contiene todas las conexiones necesarias para el control, esta se muestra en la Figura 21, de igual manera se hizo una para la conexión de la placa de radiofrecuencia.

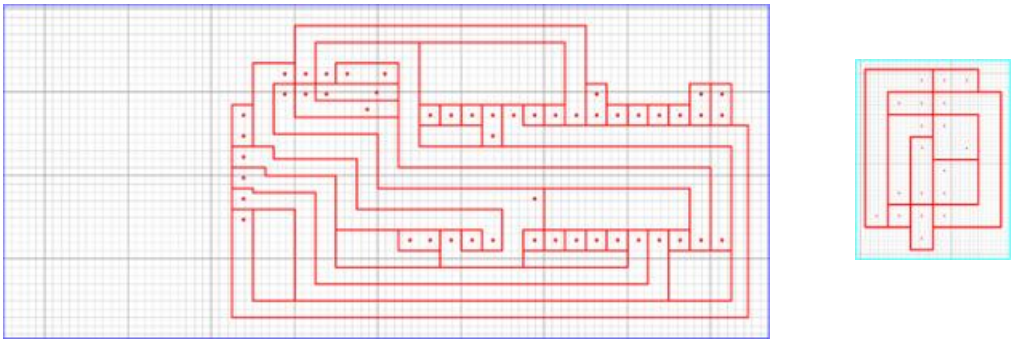


Figura 21. Placas impresas en Proteus 8, a) placa para el transmisor, b) placa para el módulo de radiofrecuencia del receptor.

Teniendo listo el prototipo se decidió caracterizar la cámara para decidir las dimensiones de las fotos y ver el ángulo de visión de esta, para esto primero se tomaron imágenes con resoluciones que fueran fracciones de las dimensiones habituales, pero se observó que la paquetería CV2 no acepta cualquier resolución, por lo que se decidió tomar la resolución permitida más pequeña, esto con el fin de agilizar el proceso de toma de datos. La resolución para las imágenes fue de 240x320 pixeles y el ángulo de visión para esta resolución fue de 56° en el eje horizontal y 45° en el eje vertical.

Una vez teniendo listo el prototipo se desarrollaron los programas necesarios para el vehículo y el control, para el receptor se hicieron dos programas uno en Arduino y uno en Python y para el transmisor se hizo un programa en Arduino. En diagrama de los programas desarrollados son los siguientes:

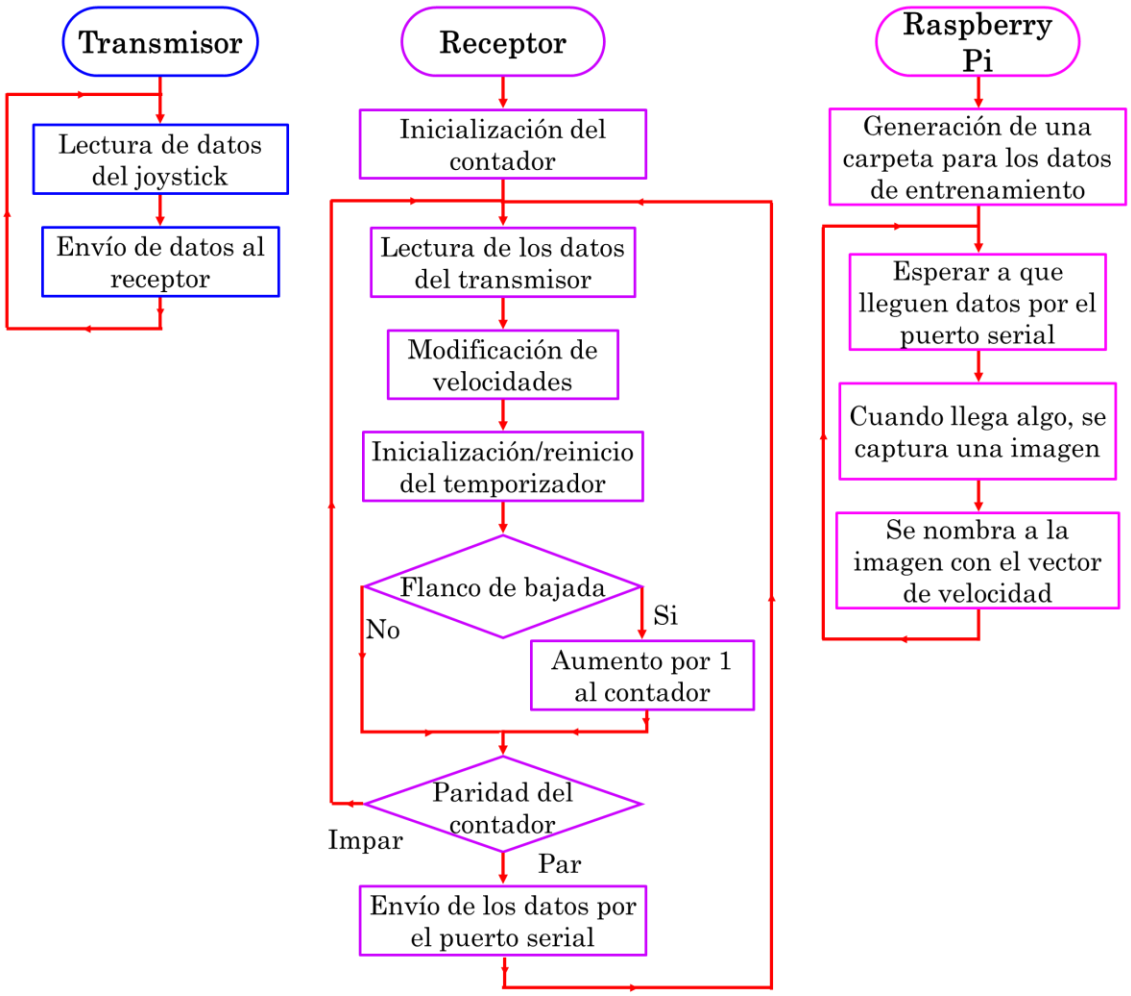


Figura 22. Programas desarrollados para el manejo del vehículo.

El programa del transmisor lo único que hace es leer los datos del joystick y los envía al receptor en todo momento, los datos que manda contendrán los datos de la posición en (X, Y) del joystick y el estado del botón.

El programa para el receptor primero habilita los motores e inicializa un contador a cero, este se usara después, inicializa o reinicia un temporizador, este determinara la rapidez con la que se leerá la información del transmisor, lee los datos provenientes del transmisor, si no hay datos, pone a los motores en estado de reposo y espera a que llegue información, una vez que llega, modifica la velocidad actual de los motores con esta información, por otra parte lee la información del botón del joystick, su valor es 1 en todo momento y cuando se oprime, su valor cambia a 0, de la Figura 23 a), se puede ver que cuando se oprime el botón pueden llegar uno o más ceros y se desea que el vehículo empiece a grabar cuando detecte el primer cero, por lo que se utilizará el flanco de bajada de la señal,

para dejar de grabar es lo mismo, se utilizará el flanco de bajada de la señal, es aquí donde entra el contador, cada vez que se detecta un flanco de bajada, el contador aumenta por 1 y se pregunta por su paridad, si es impar, se mandan los datos por el puerto serial, de esta manera se empezará a grabar los datos de entrenamiento, después se regresa a la lectura de los datos y se repite el proceso, si es par, no se manda información y simplemente se repite el proceso.

Y, por último, el programa para la Raspberry primero genera una carpeta que es la que contendrá todos los datos de entrenamiento, cada vez que se corra el programa, se generará una carpeta nueva, una vez hecho esto, espera a que llegue información por el puerto serial, si no hay información, no hace nada, una vez que llega información empieza a tomar imágenes, las enumera y las guarda junto con la información que le llega por el puerto serial, la información que llega se verá como se muestra en la Figura 23.

Figura 23. a) Datos generados por el transmisor, b) datos recibidos por la Raspberry Pi, c), d) datos de entrenamiento generados.

Este sistema parecía funcionar muy bien, pero surgió un problema, el motor de la llanta derecha tenía un comportamiento extraño, solo se activaba con un valor de digitalización de PWM de 255 y para valores más pequeños que eso, el motor se quedaba inmóvil, este era un problema ya que, si se quisiera mover el vehículo hacia adelante, la llanta izquierda empezaría a girar y la derecha se quedaría inmóvil hasta alcanzar el valor de 255, esto produce que el vehículo gire sobre la llanta derecha.

Este problema se intentó atacar desde distintos ángulos y al final se pudo solucionar modificando el programa en Arduino, ya que la librería para inicializar el temporizador que sería usado para hacer las interrupciones usadas para agilizar el programa, también se utilizaban para mover el motor derecho y esto

producía que el PWM que le llegaba fuera incorrecto, por lo que se decidió utilizar los flancos de bajada como en el programa de la Figura 22.

Una vez teniendo el prototipo listo, se decidió empezar a tomar los datos de entrenamiento, pero se presentó otro problema, el vehículo presentaba dificultades para dar las vueltas, esto se debía a que los motores no tenían la potencia necesaria para conducir el vehículo, ya que para dar vueltas se debe producir un pequeño incremento o decremento en la señal de PWM pero la señal necesaria para vencer la inercia de los motores mismos es a partir de una digitalización de PWM de 60, entonces, cuando se quería dar una vuelta abierta, el vehículo no las daba y cuando al fin se alcazaba el valor necesario, el vehículo daba una vuelta muy cerrada. Este es un problema ya que no se podía corregir de manera fina la dirección del vehículo, este problema se puede corregir de dos formas, una es utilizando motores más potentes para el manejo del prototipo y la otra forma es cambiar el mecanismo del vehículo para un mejor manejo.

Al final se decidió solucionar este problema cambiando la estructura del prototipo a una estructura similar a la de un coche convencional, para esto se utilizará un motor más potente para la tracción de las llantas traseras y un pequeño servomotor que se encargará de dar dirección al vehículo, de esta manera se puede modificar de manera precisa la dirección ya que el servomotor no necesita mucha potencia para trabajar, solo se necesita cambiar el tiempo en alto de la señal que se le manda.

Con esta idea se desarrolló un nuevo prototipo, al principio se empezó a desarrollar el vehículo desde cero, creando las piezas mecánicas en impresión 3D pero a la hora de hacer esto, las piezas quedan escalonadas debido a que la impresora genera las piezas por capas, al hacer esto, podrían surgir posibles problemas a la hora de imprimir el engranaje para mover las llantas, además de que el desarrollo de este prototipo llevaría tiempo, por lo que se decidió buscar un kit que se adecuara a las necesidades del proyecto.

2.2.4 Cuarto prototipo

Para la parte mecánica de este prototipo se utilizó un JetRacer Pro AI Kit que es un modelo de control remoto diseñado para trabajos de inteligencia artificial, este ya cuenta con una Cámara para grabar su recorrido.

El mecanismo de este prototipo es similar al de un coche tradicional, por una parte, tiene un motor que se encarga de dar tracción a las llantas traseras y un servomotor que se encarga de darle dirección a las llantas delanteras, este prototipo cuenta con un diferencial en la parte trasera y delantera, este se encarga de distribuir la energía a ambas llantas, éste es útil a la hora de dar vueltas ya que se evita que las llantas se patinen, este prototipo también cuenta con tracción 4x4 y una serie de amortiguadores.

El JetRacer Pro AI Kit, fue diseñado para trabajar con la Jetson Nano, por lo que, en un principio se intentó usar esta tarjeta que cuenta con una tarjeta gráfica que disminuye el tiempo de procesamiento de las imágenes, por lo que parecía ser más eficiente que la Raspberry Pi, sin embargo, surgían problemas a la hora de querer usar las librerías necesarias para la toma de imágenes en Python, esto se debía a que algunas librerías no eran compatibles con la tarjeta gráfica, debido a esto, se intentó instalar versiones que fueran compatibles con esta pero surgían otros problemas por lo que se decidió instalar el sistema operativo que ya tiene todas las librerías necesarias para el uso de este kit, pero se observó que este sistema era muy lento, parecería que el sistema estuviera ejecutando tareas innecesarias, debido a esto, se decidió utilizar este kit con la Raspberry pi y con los programas que ya se habían desarrollado anteriormente, para esto, se modificaron ligeramente los programas.

Figura 24. Estructura del JetRacer Pro AI Kit.

En el siguiente capitulo se mostrarán los resultados de la implementación de las tres técnicas de análisis de datos aplicadas a los datos de entrenamiento generados con este prototipo.

CAPÍTULO 3. RESULTADOS EXPERIMENTALES

Para el análisis de los datos de entrenamiento del proyecto se utilizaron los tres métodos mencionados anteriormente, la técnica de Proyección de Estructuras Latentes, la de Regresión de Componentes Principales, y una Red Neuronal Convolucional seguida por una red de regresión. De esta manera se obtuvo una predicción del ángulo que debería tener el vehículo en todo momento y para calcular el desempeño de los modelos generados mediante estas técnicas se calculó el error absoluto promedio entre los valores predichos y los deseados para los distintos datos de entrenamiento, así como su desviación estándar.

La pista utilizada para obtener los datos se muestra en la Figura 25

Figura 25. Pista de prueba utilizada para obtener los datos de entrenamiento.

Para este análisis se estudiaron cinco conjuntos de datos de entrenamiento con distinto número de imágenes, en la Tabla 1 se muestra el número de elementos de cada conjunto.

Tabla 1. Conjuntos de datos analizados.

Conjunto	Número de datos
Conjunto 1	11000
Conjunto 2	5400
Conjunto 3	2000
Conjunto 4	5300
Conjunto 5	3500

A continuación, se mostrará la implementación de estas técnicas de análisis y se discutirán sus resultados.

3.1 RESULTADOS DE LA IMPLEMENTACION DEL METODO DE PLS-R

El entrenamiento del modelo se realizó aplicando la tecnica de PLS-R a los conjuntos de datos de la Tabla 1, utilizando el proceso de validación. De esta manera se obtuvo la desviación estandar de la salida de los datos de validacion y se calculó el numero de componentes que generaba la minima desviación, este proceso se hizo con tres resoluciones diferentes, en la Figura 26 se muestra el resultado obtenido con el conjunto 1.

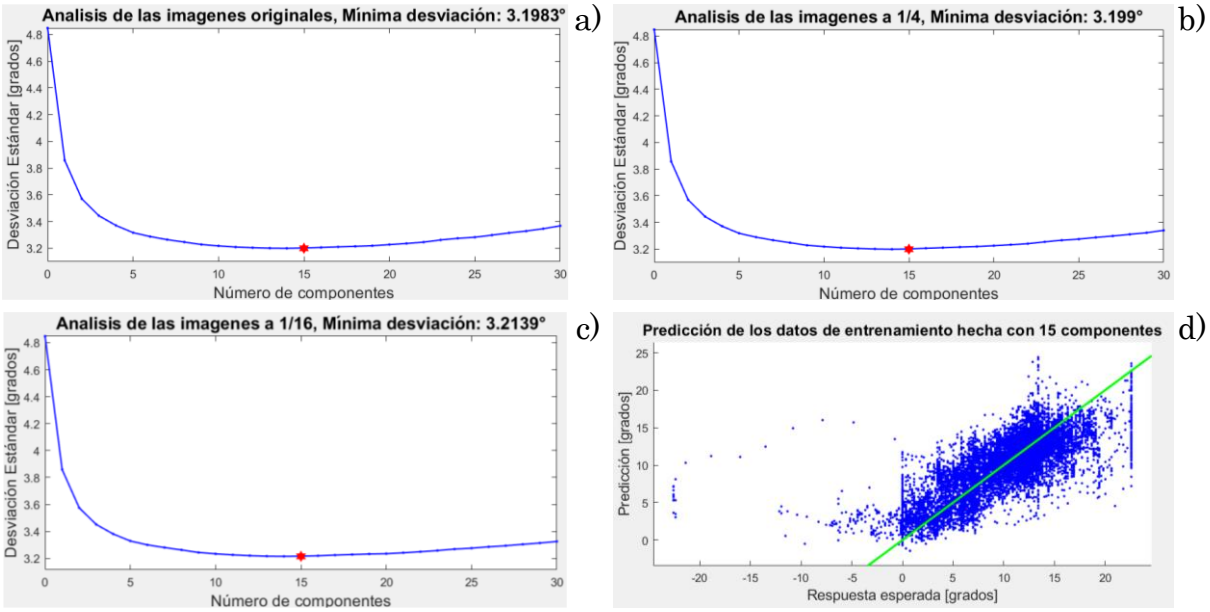


Figura 26. Desviación estándar de los datos de validación en función del número de componentes para las imágenes con resoluciones de: a) 240x320 pixeles, b) 120x160 pixeles y c) 60x80 pixeles. d) Respuesta esperada VS la predicha, la línea verde indica la relación "ideal".

Como se puede observar, la reducción de dimensiones no afecta al rendimiento del modelo de manera significativa, por lo que se pueden utilizar imágenes de 60x80 pixeles para generar los modelos. Por otra parte, la desviación estándar más pequeña es del orden de 3.2°y en general se obtiene con 15 componentes. Mas allá de este número la desviación estándar aumenta, sugiriendo un proceso de sobreajuste con los datos de entrenamiento. La Figura 26d muestra los resultados de las predicciones para este conjunto de datos VS los ángulos deseados, como se observa, el desempeño no es el mejor ya que en algunas

regiones se separa mucho de la línea recta, por lo que fue necesario analizar más datos de entrenamiento para obtener un mejor resultado.

Todos los datos de la Tabla 1 fueron analizados y se ordenaron del menor al mayor desempeño obtenido, con la idea de mostrar la calidad obtenida con estos conjuntos de datos y no ser repetitivos, solo se muestran los análisis de tres grupos de datos, el conjunto 1, 3 y 5. En la Figura 27 se muestran los resultados obtenidos al analizar el conjunto 3.

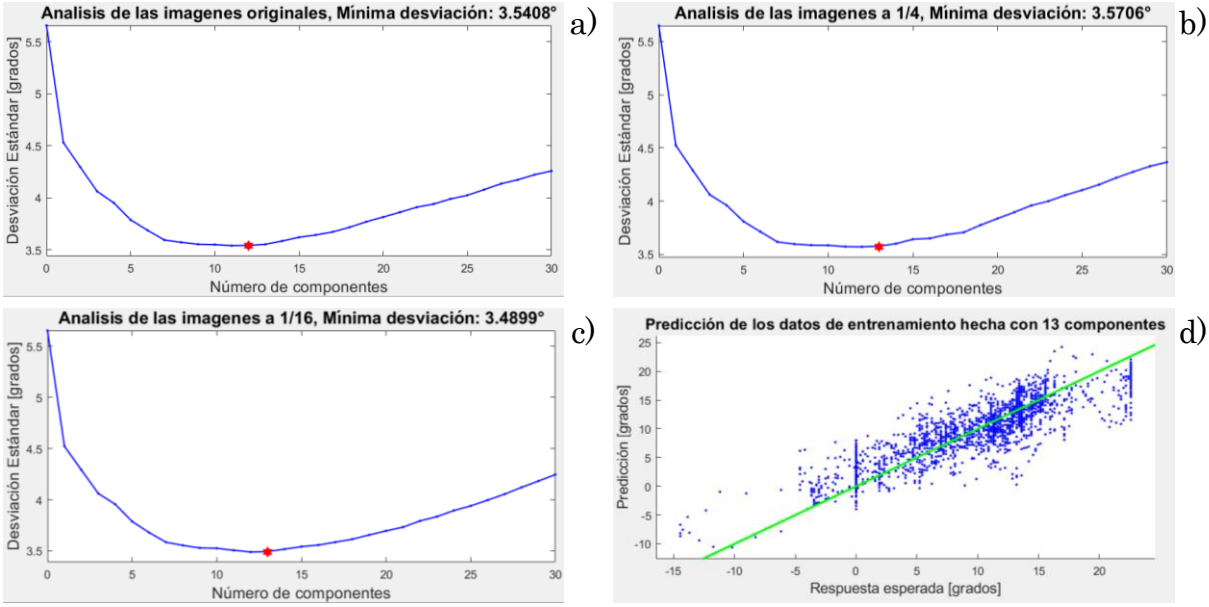


Figura 27. Desviación estándar de los datos de validación en función del número de componentes para las imágenes con resoluciones de: a) 240x320 pixeles, b) 120x160 pixeles y c) 60x80 pixeles. d) Respuesta esperada VS la predicha, la línea verde indica la relación ideal.

Se puede observar que con este conjunto de datos de entrenamiento existe una pequeña mejoría en el número de componentes necesarias, ahora solo se requieren 14. Por otra parte, también se observa que el sobreajuste se manifiesta de manera más rápida al tomar un mayor número de componentes. Los resultados del análisis del conjunto de datos 5 se muestran en la Figura 28.

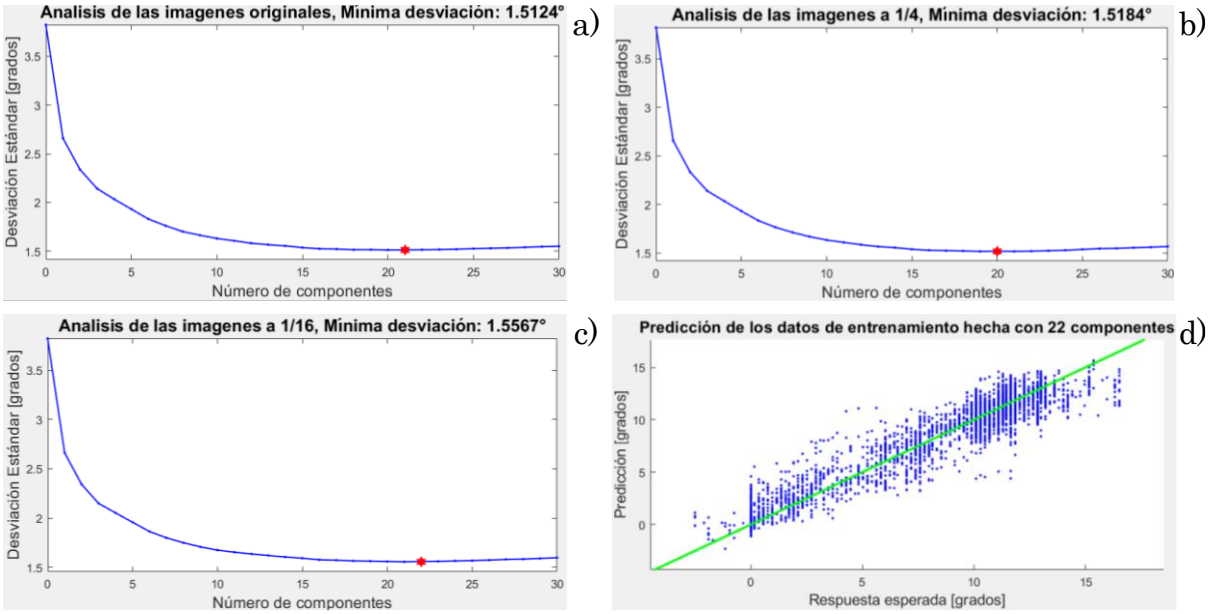


Figura 28. Desviación estándar de los datos de validación en función del número de componentes para las imágenes con resoluciones de: a) 240x320 pixeles, b) 120x160 pixeles y c) 60x80 pixeles. d) Respuesta esperada VS la predicha, para el conjunto de datos 5.

Se puede observar que para este conjunto de datos se tiene el mejor resultado, en este caso la desviación estándar es de 1.5° con el uso de 22 componentes. Esto puede confirmarse en la Figura 28d en donde las predicciones presentan un menor grado de esparcimiento alrededor de la línea ideal.

De los resultados anteriores se puede ver que la disminución de las dimensiones de las imágenes analizadas no afecta de manera significativa el rendimiento de los modelos, por esta razón se decidió trabajar con las imágenes de menor resolución. Esto permite disminuir el número de operaciones para entrenar los modelos, lo que significa una reducción en el coste computacional, además de reducir el tiempo de cómputo para realizar predicciones de nuevos datos. Por lo que para los siguientes análisis de los datos se utilizarán las imágenes con una resolución de 60x80 pixeles (una reducción de 1/16 en la resolución original). Es importante mencionar que para este proceso de reducción de dimensiones se utilizó un promedio con los pixeles vecinos para generar la reducción.

Una vez que se vio este comportamiento, se realizó el entrenamiento y la prueba con distintos conjuntos de datos, se evaluó el desempeño de éstos, en función del número de componentes con la idea de obtener una mejor generalización.

En la Figura 29 se muestra la curva de entrenamiento usando PLS-R para el conjunto 1 como datos de entrenamiento y el conjunto 2 como datos de prueba.

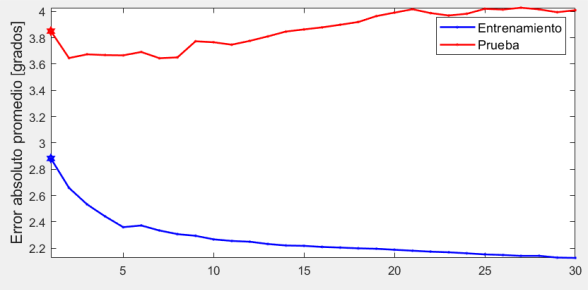


Figura 29. Comparación del error absoluto promedio entre las respuestas esperadas y predichas para los datos analizados usando distinto número de componentes en el análisis de PLS-R.

Se puede observar que la separación entre las curvas inicialmente es grande y crece al aumentar el número de componentes, lo que significa que el entrenamiento adolece de un fuerte sobreajuste (los datos de entrenamiento y de prueba son muy diferentes). Otra forma de observar este comportamiento se presenta en la Figura 30. Aquí se muestra la relación entre la predicción y la respuesta esperada para los datos de entrenamiento y prueba, respectivamente cuando se utiliza una sola componente de PLS.

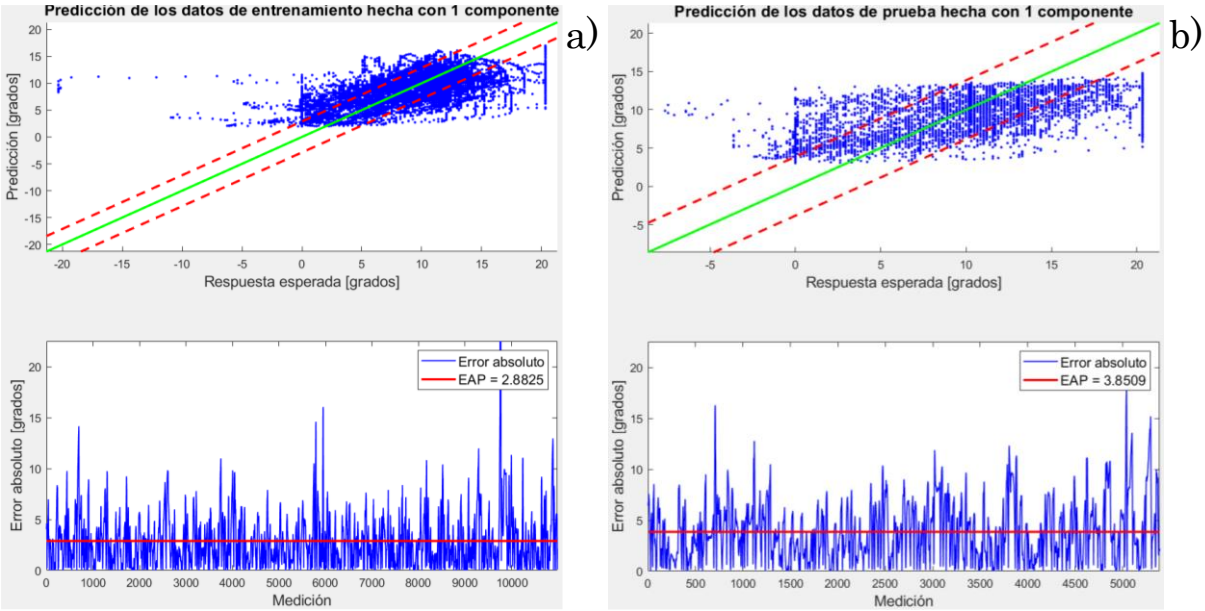


Figura 30. Comparación de la salida predicha por el modelo utilizando 1 componente contra la respuesta esperada para los datos analizados, en a) están los resultados para los datos de entrenamiento y en b) los de los datos de prueba, en la parte inferior se muestra el error absoluto de las predicciones y el promedio de estos errores.

Como era de esperarse el error es grande y se puede observar que la tendencia de estos datos se asemeja más a una recta horizontal que a la curva deseada, lo que produce que el error absoluto promedio sea relativamente grande.

Se puede observar que en algunas mediciones el error supera por mucho este promedio, esto sucede principalmente cuando la respuesta esperada es menor a cero (vuelta a la izquierda y ángulos de la dirección negativos), esto se debe a que la trayectoria seguida para obtener los datos de entrenamiento y prueba solo contiene vueltas positivas (a la derecha), por lo que, para tener una mejor predicción en estas regiones, es necesario tener una pista de pruebas más complicada, con curvas hacia ambos lados.

Con la idea de mostrar con mayor claridad la diferencia entre los errores en ambos conjuntos de datos se calculó el histograma de los errores, para ver su dispersión y una gráfica que muestra las respuestas deseadas y predichas por el modelo para ambos conjuntos, estos resultados se muestran en la Figura 31.

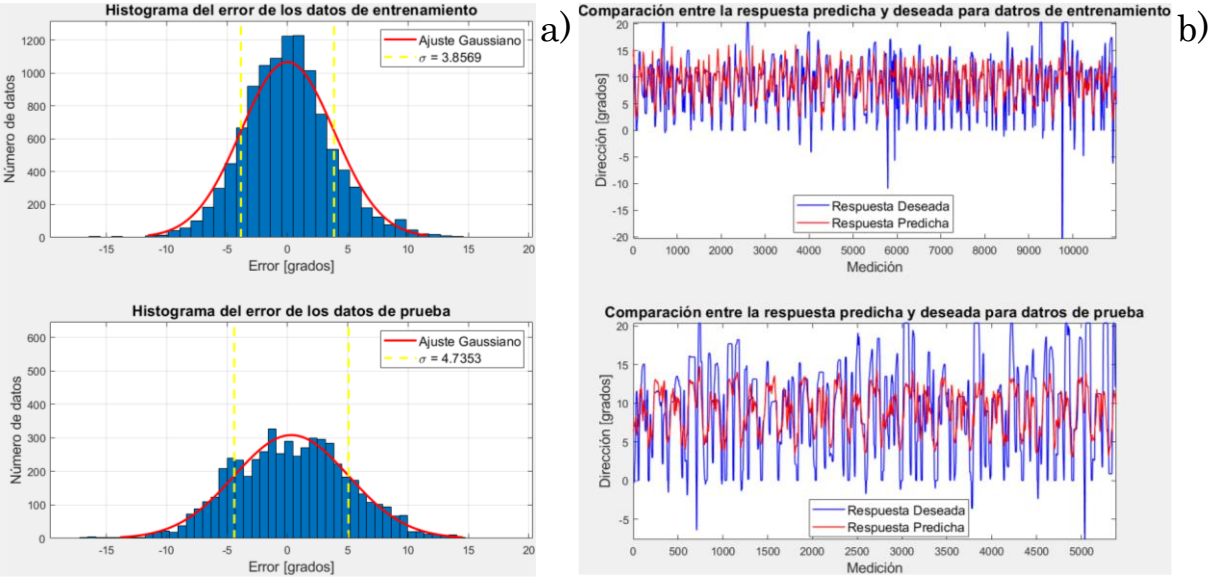


Figura 31. En a) se muestran los histogramas del error para los datos de entrenamiento y de prueba (arriba y abajo, respectivamente), también se presenta un ajuste gaussiano y la región donde se concentra la mayor parte de estos datos. b) comparación entre las respuestas deseadas y predichas para estos conjuntos de datos.

Se puede observar que el error de los datos de prueba está más disperso, comparado con los errores de los datos de entrenamiento, lo que se refleja en un

comportamiento ruidoso en las predicciones. Esto se debe probablemente a un sobreajuste debido a las diferencias presentes en ambos conjuntos.

Por otra parte, también se analizaron los conjuntos 2 y 4 como entrenamiento y prueba, respectivamente, estos resultados se muestran en la Figura 32.

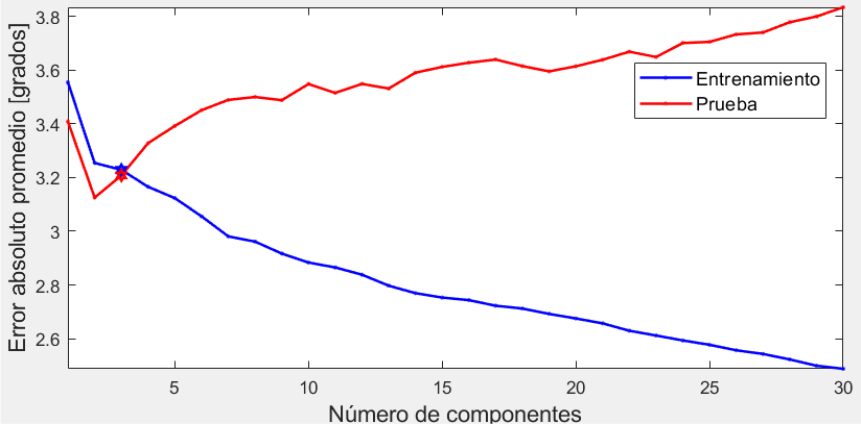


Figura 32. Comparación del error absoluto promedio entre las respuestas esperadas y predichas para los datos analizados usando distinto número de componentes en el análisis de PLS-R.

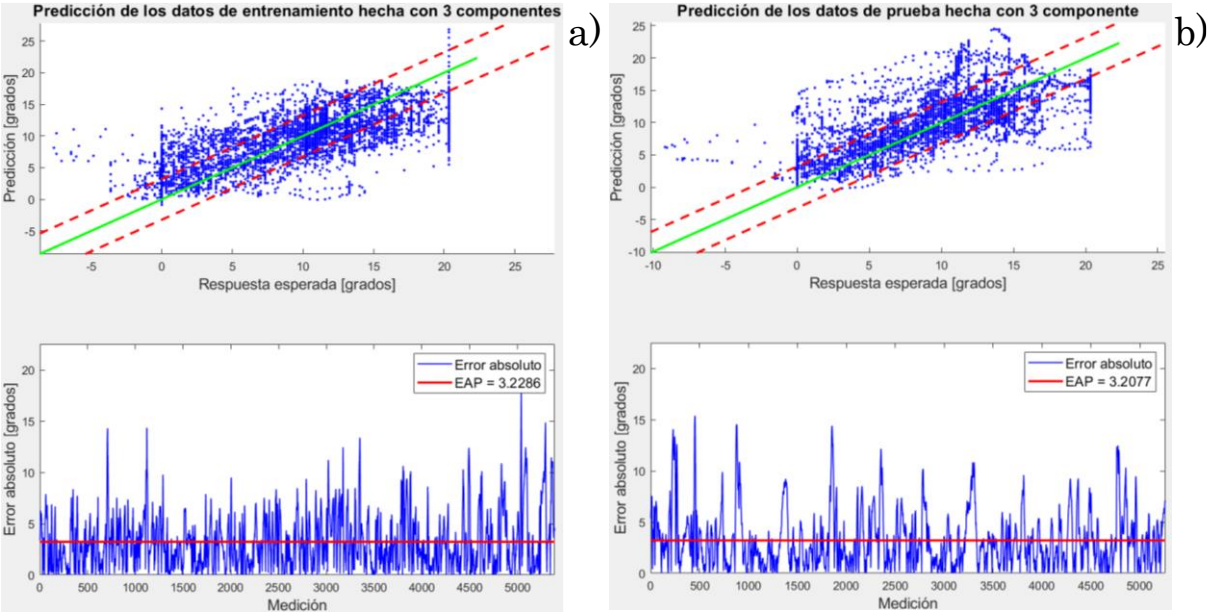


Figura 33. Comparación de la salida predicha por el modelo utilizando 3 componente contra la respuesta esperada para los datos analizados, en a) están los resultados para los datos de entrenamiento y en b) los de los datos de prueba, en la parte inferior se muestra el error absoluto de las predicciones y el promedio de estos errores.

El error absoluto para este caso es ahora del mismo orden en ambos conjuntos de datos, cuando se utilizan 3 componentes, y la diferencia en ambos aumenta a medida que se utilizan más componentes. Como se observa en las gráficas superiores de la Figura 33 tendencia de las predicciones en los datos de entrenamiento y prueba se asemejan más a la recta deseada.

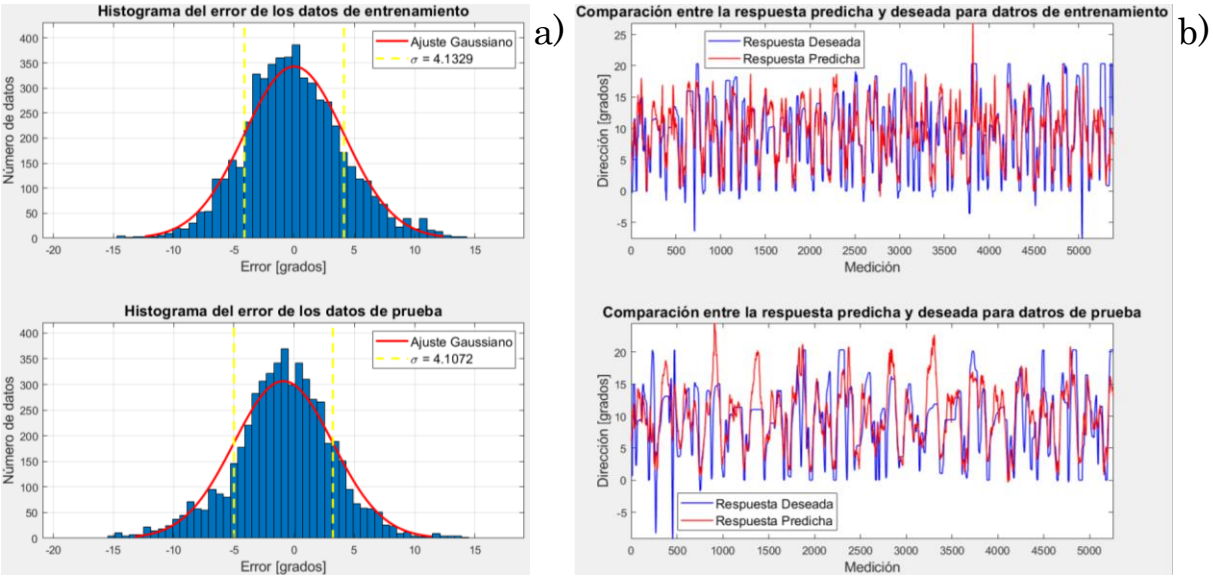


Figura 34. En a) se muestran los histogramas del error para los datos de entrenamiento y de prueba (arriba y abajo, respectivamente), también se presenta un ajuste gaussiano y la región donde se concentra la mayor parte de estos datos. En b) se muestra una comparación entre las respuestas deseadas y predichas para estos conjuntos de datos.

Los histogramas de la Figura 34 corroboran esta similitud, en donde la desviación estándar para los datos de entrenamiento y prueba fueron de 4.13° y 4.1°, respectivamente. Esta mejoría también se observa en las gráficas de la derecha de la Figura 34.

Por otra parte, el análisis de los conjuntos 4 y 5 como entrenamiento y prueba se presentan en la Figura 35.

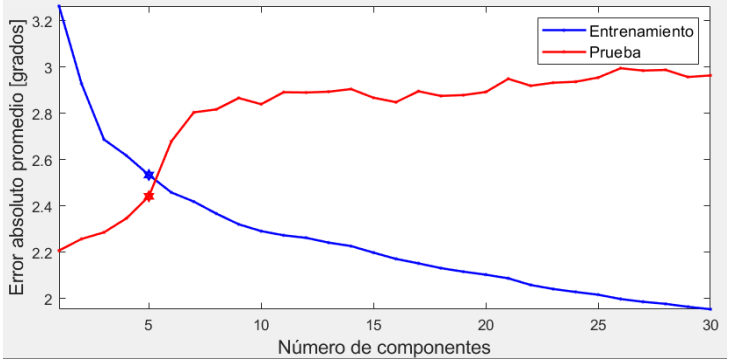


Figura 35. Comparación del error absoluto promedio entre las respuestas esperadas y predichas para los datos analizados usando distinto número de componentes en el análisis de PLS-R.

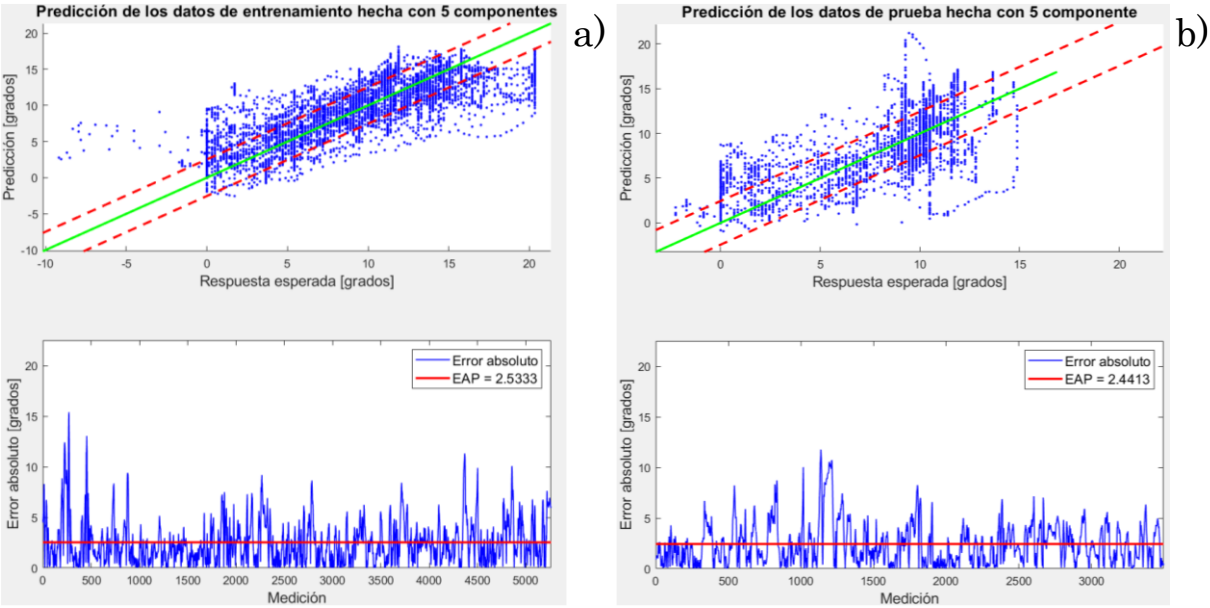


Figura 36. Comparación de la salida predicha por el modelo utilizando 5 componente contra la respuesta esperada para los datos analizados, en a) están los resultados para los datos de entrenamiento y en b) los de los datos de prueba, en la parte inferior se muestra el error absoluto de las predicciones y el promedio de estos errores.

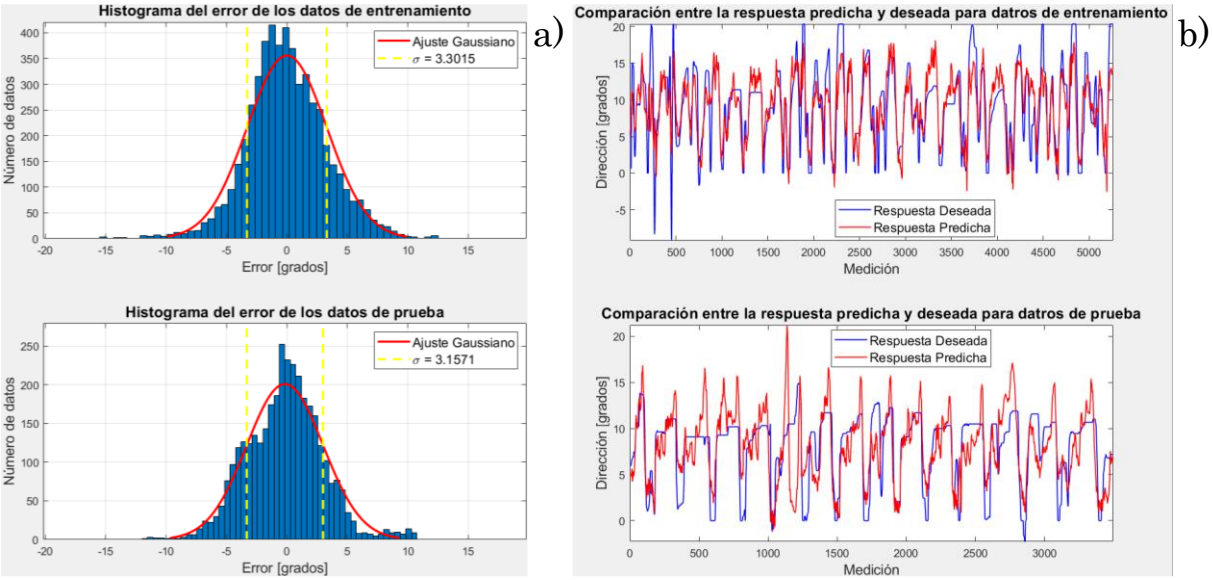


Figura 37. En a) se muestran los histogramas del error para los datos de entrenamiento y de prueba (arriba y abajo, respectivamente), también se presenta un ajuste gaussiano y la región donde se concentra la mayor parte de estos datos. En b) se muestra una comparación entre las respuestas deseadas y predichas para estos conjuntos de datos.

Como era de esperarse, la discrepancia entre los errores en los datos de entrenamiento y prueba es menor. En este caso los errores para los datos de entrenamiento y prueba son de 2.53° y 2.44°, respectivamente cuando se utilizan 5 componentes. Esto se confirma con los histogramas y la comparación entre la respuesta predicha y la deseada de la Figura 37, particularmente en esta última puede notarse que las predicciones ahora muestran valores cercanos a cero, pero aun es incapaz de predecir los valores de ángulos negativos.

En términos generales, se puede observar que estos errores siguen siendo relativamente grandes, esto puede deberse a la calidad de los datos analizados junto con la necesidad del uso de otras técnicas de análisis para estos datos. A continuación, se presentan los resultados con el uso de la técnica de PCR con la misma estrategia de análisis mostrada con PLS.

3.2 RESULTADOS DE LA IMPLEMENTACION DEL METODO DE PCR

Al igual que en la sección anterior, la reducción de las dimensiones de las imágenes analizadas no afecta al rendimiento de los modelos generados con PCR,

por lo que, los siguientes análisis se realizaran con las imágenes reducidas a 1/16 de la resolución original.

De igual forma, se realizó el entrenamiento y la prueba de estos modelos con los distintos conjuntos de la Tabla 1, primero se obtuvieron las componentes principales (PC's) y después se aplicó la regresión (PCR) con los primeros n componentes, de esta manera se calculó el número de componentes con el que se obtiene una mejor generalización.

En la Figura 38 se muestran los resultados del análisis con PCR de los conjuntos 1 y 2 como entrenamiento y prueba, respectivamente.

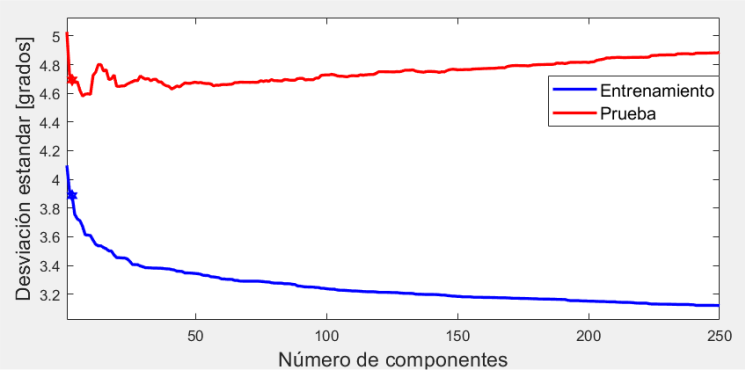


Figura 38. Comparación del error absoluto promedio entre las respuestas esperadas y predichas para los datos analizados usando distinto número de componentes en el análisis con PCR.

Como se puede observar, al analizar estos conjuntos de datos, se obtiene un resultado similar al obtenido con la técnica de PLS-R, la diferencia entre las curvas generadas con los datos de entrenamiento y prueba inicialmente es grande y aumenta cuando crece el número de componentes, por lo que también se tiene un sobreajuste. De igual forma, para ver el desempeño de este modelo se comparó la respuesta deseada y predicha para estos conjuntos, en la Figura 39 se muestran estos los resultados.

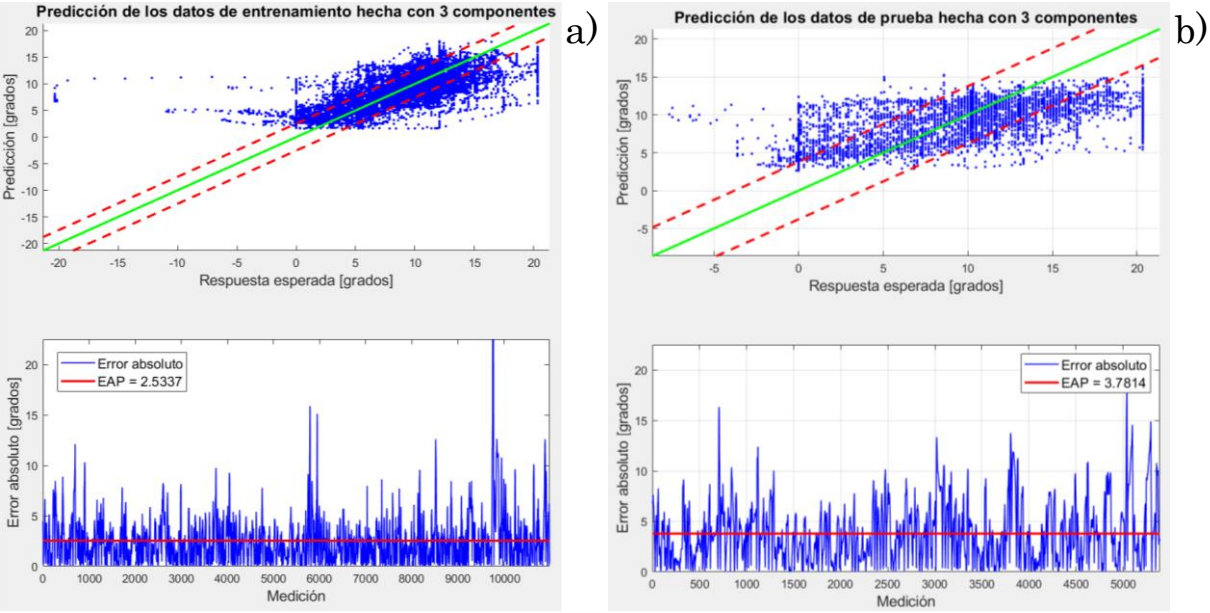


Figura 39. Comparación de la salida predicha por el modelo utilizando 3 componentes contra la respuesta esperada para los datos analizados, en a) se muestran los resultados para los datos de entrenamiento y en b) los de los datos de prueba, en la parte inferior se muestra el error absoluto de las predicciones y el promedio de estos errores.

Estos resultados tienen un comportamiento similar al obtenido con PLS-R, sin embargo, se puede observar que el error absoluto promedio para el análisis con PCR es menor que el obtenido con PLS-R, con una diferencia del orden de 0.35° para el caso de los datos de entrenamiento y 0.07° para los datos de prueba, esta diferencia no es lo suficientemente grande, por lo que para tener una mejor comparación, se calculó el histograma del error obtenido con el método de PCR y se comparó las respuestas deseadas y predichas para los conjuntos analizados, estos resultados se muestran en la Figura 40.

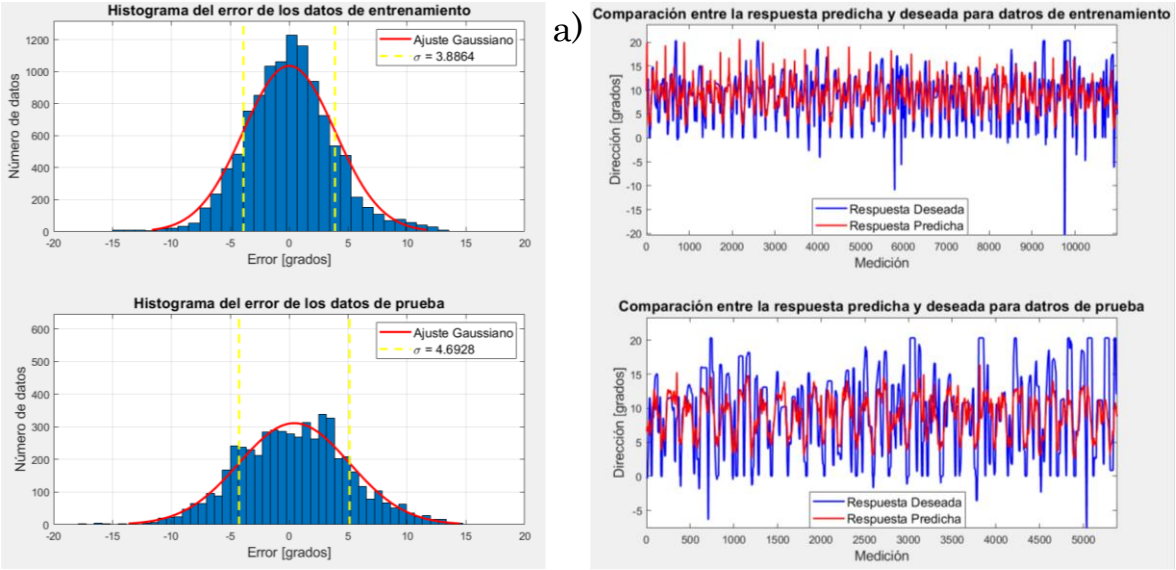


Figura 40. En a) se muestran los histogramas del error para los datos de entrenamiento y de prueba (arriba y abajo, respectivamente), también se presenta un ajuste gaussiano y la región donde se concentra la mayor parte de estos datos. En b) se muestra una comparación entre las respuestas deseadas y predichas para estos conjuntos de datos.

Se puede observar que en comparación con los resultados obtenidos con PLS, la dispersión del error para los datos de entrenamiento es mayor, con un orden de 0.03°, mientras que para el conjunto de prueba es menor, con un orden de 0.04°, esta diferencia no es significativa, se puede observar que se sigue teniendo el mismo comportamiento que el obtenido con PLS-R, cuando el ángulo es menor a cero, la predicción se aleja mucho de la respuesta desaseada para ambos grupos, además, ambos modelos presentan un sobreajuste al analizar estos conjuntos de datos por lo que se puede llegar a la conclusión de que estos datos son malos, esto se puede deber a una gran diferencia de la información de ambos grupos.

Por otra parte, se analizaron los conjuntos 2 y 4 como entrenamiento y prueba, respectivamente, estos resultados se muestran en la Figura 41.

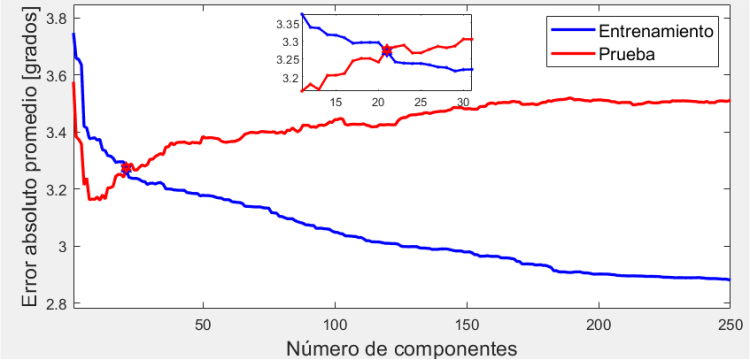


Figura 41. Comparación del error absoluto promedio de las respuestas esperadas y predichas para los datos de entrenamiento y prueba usando distinto número de componentes para hacer PCR, donde se resalta el número de componentes con los que hay una mejor generalización.

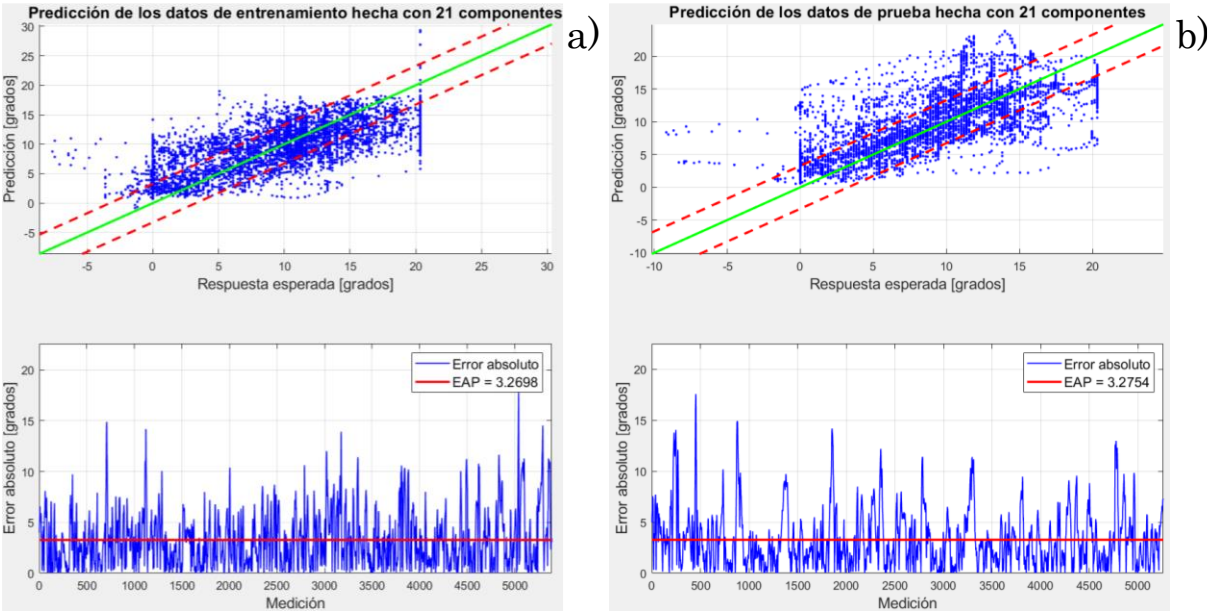


Figura 42. Comparación de la salida predicha por el modelo utilizando 1 componente contra la respuesta esperada para los datos analizados, en a) están los resultados para los datos de entrenamiento y en b) los de los datos de prueba, en la parte inferior se muestra el error absoluto de las predicciones y el promedio de estos errores.

Se puede observar, se tiene una mejor generalización cuando se utilizan 21 componentes para hacer PCR, mientras que para PLS-R se obtuvo con 3 componentes, esto se puede deber a que PCR calcula los componentes con base a la varianza de los datos de entrada, mientras que PLS-R calcula los componentes con base a los datos de entrada y salida. Por otra parte, se puede observar una disminución en el error absoluto promedio, con respecto al análisis de los conjuntos

anteriores, por lo que se puede decir que estos conjuntos de datos son más similares entre sí que los analizados anteriormente. El histograma del error para estos datos se muestra en la Figura 43.

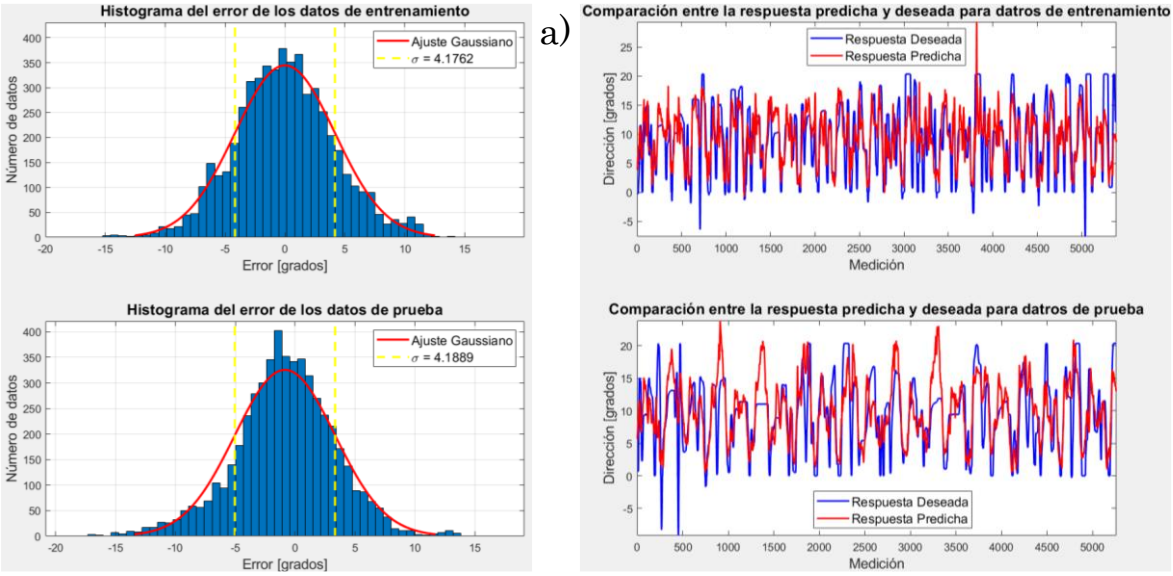


Figura 43. En a) se muestran los histogramas del error para los datos de entrenamiento y de prueba (arriba y abajo, respectivamente), también se presenta un ajuste gaussiano y la región donde se concentra la mayor parte de estos datos. En b) se muestra una comparación entre las respuestas deseadas y predichas para estos conjuntos de datos.

Del histograma de la figura anterior, se puede observar que el error para los datos de entrenamiento y prueba tienen una dispersión similar, con una desviación estándar del orden de 4.1° y en la gráfica de la derecha, se puede observar que la predicción para los datos de prueba y entrenamiento se acerca más a la respuesta esperada para estos datos.

Por último, se analizaron los dos conjuntos de datos que parecen ser los mejores (conjunto 4 y 5), los resultados de este análisis se muestran en la Figura 44.

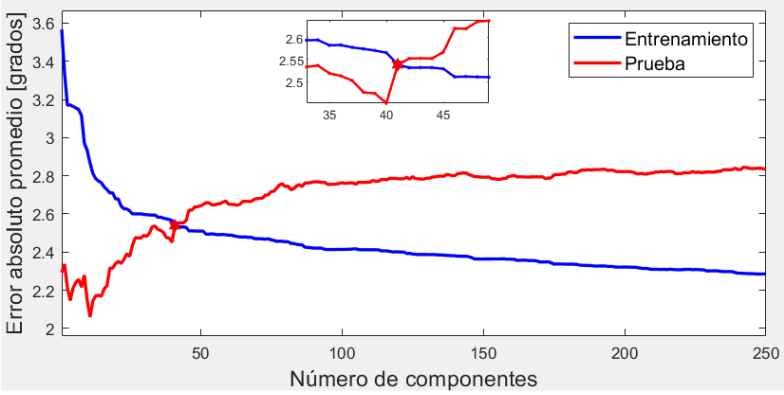


Figura 44. Comparación del error absoluto promedio de las respuestas esperadas y predichas para los datos analizados usando distinto número de componentes para hacer PCR, donde se resalta el número de componentes con los que hay una mejor generalización.

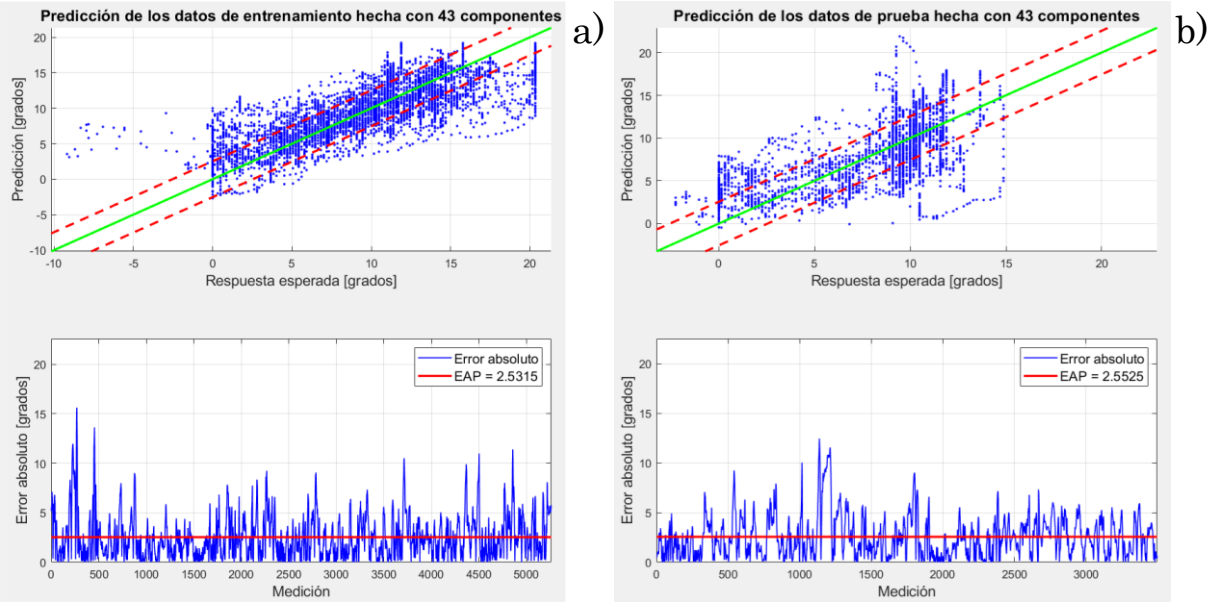


Figura 45. Comparación de la salida predicha por el modelo utilizando 1 componente contra la respuesta esperada para los datos analizados, en a) están los resultados para los datos de entrenamiento y en b) los de los datos de prueba, en la parte inferior se muestra el error absoluto de las predicciones y el promedio de estos errores.

Se puede observar que, para estos conjuntos de datos, se obtiene una mayor generalización al utilizar 43 componentes, logrando así un error absoluto promedio del orden de 2.5° para los conjuntos de entrenamiento y prueba. Lo que confirma que estos dos conjuntos de datos son los más compatibles entre sí. El histograma del error de estos datos se muestra en la Figura 46.

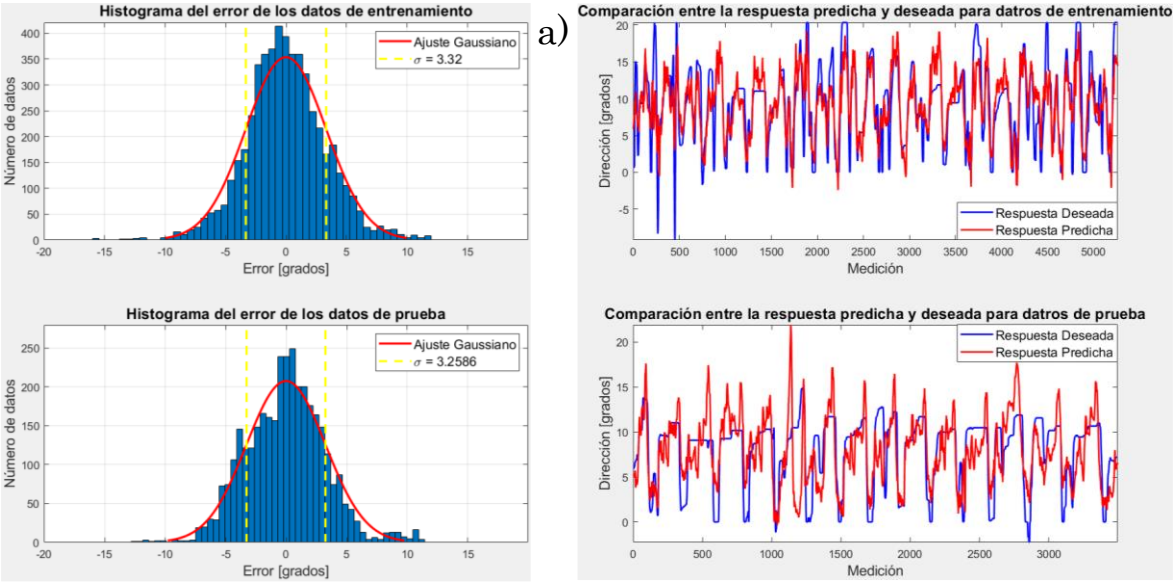


Figura 46. En a) se muestran los histogramas del error para los datos de entrenamiento y de prueba (arriba y abajo, respectivamente), también se presenta un ajuste gaussiano y la región donde se concentra la mayor parte de estos datos. En b) se muestra una comparación entre las respuestas deseadas y predichas para estos conjuntos de datos.

Como era de esperarse, se tiene una menor dispersión del error para estos conjuntos de datos, con valores de 3.32° para el conjunto de entrenamiento y 3.25° para el de prueba. Además, se puede observar que las predicciones siguen de mejor manera los valores deseados.

Se puede observar un aumento en el número de componentes utilizados con el método de PCR, con respecto al método de PLS, esto se debe a que el método de PCR calcula las componentes con base a la varianza de los datos de entrada, mientras que el método de PLS-R lo hace con base a los datos de entrada y salida, por lo que contienen más información relevante para hacer las predicciones de las salidas.

Se puede observar que los modelos obtenidos con PCR y PLS-R no dan una buena predicción cuando se tienen direcciones negativas, esto se debe principalmente a que la trayectoria seguida para obtener los datos de entrenamiento y prueba es ovalada, por lo que solo se pueden dar vueltas en un sentido (en este caso positivo). Por lo que es necesario el uso de una pista más elaborada para la toma de datos.

Los resultados obtenidos mediante el uso de PLS-R son ligeramente mejores que los obtenidos con la técnica de PCR, sin embargo, estos resultados siguen presentando un error relativamente grande, por lo que se analizarán estos datos con una Red Neuronal esperando obtener un mejor resultado.

3.3 RESULTADOS DE LA IMPLEMENTACIÓN DE UNA RED NEURONAL

Para este trabajo se desarrolló la red neuronal mostrada en la Figura 47, esta está compuesta por una red de convolución como extractor de características y una red de regresión que es la que dará como resultado una predicción de la dirección del modelo.

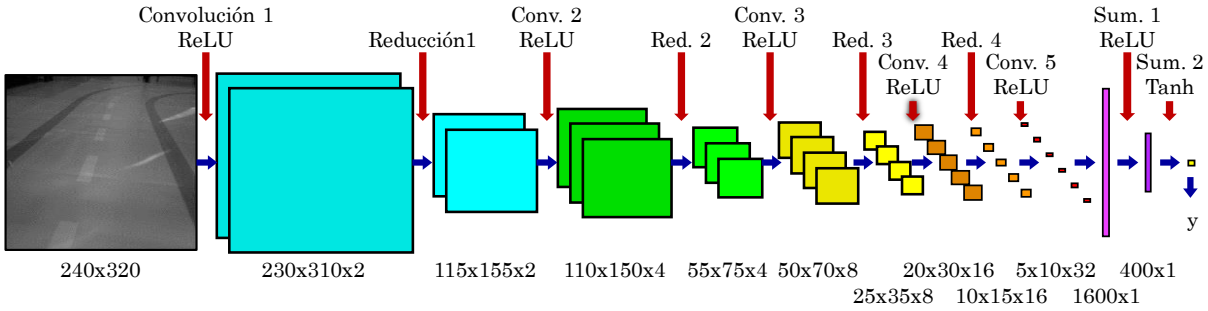


Figura 47. Estructura de la Red Neuronal usada para el análisis de las imágenes obtenidas por el KITT.

La entrada de esta red son imágenes en escala de grises del entorno del vehículo con resolución de 240x320 pixeles, la red de convolución está compuesta por cinco capas de filtros digitales y 4 capas de reducción, produciendo así 32 matrices de 5x10 pixeles, después estas matrices son reagrupadas como un vector columna con 1600 términos, este vector es entregado a una red de regresión que contiene una capa oculta con 400 nodos y un nodo de salida que corresponde a la dirección de las llantas del vehículo. Esta red fue desarrollada en TensorFlow.

Como se vio en los análisis anteriores, los conjuntos de datos con los que se tiene un mejor desempeño son con el conjunto 4 como entrenamiento y el 5 como datos de prueba. Por lo que para el análisis con la red neuronal de convolución (RNC) se utilizarán directamente estos conjuntos. En la Figura 48 se muestra el error absoluto promedio y el error cuadrático medio obtenidos durante el entrenamiento del modelo, estos resultados son para los conjuntos de formación y validación.

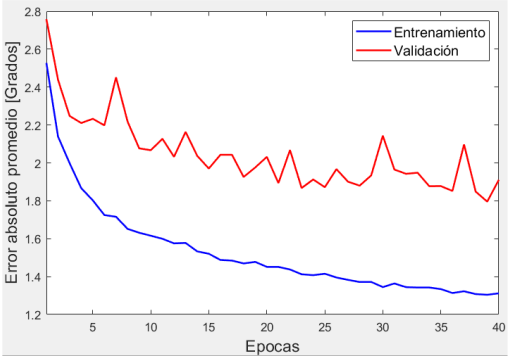


Figura 48. Entrenamiento de la red neuronal, en esta se muestra el error absoluto promedio entre la salida de los datos de formación y de validación.

Se puede observar que el error absoluto promedio para el conjunto de formación está disminuyendo "rápidamente", mientras que, para el conjunto de validación estos están disminuyendo lentamente y presentan fluctuaciones significativas, lo que provoca que estos resultados se alejen más en cada época, por lo que se puede decir que se tiene un sobreajuste al utilizar esta red.

En la Figura 49 se muestran los resultados de la implementación de este modelo a el conjunto de entrenamiento y prueba.

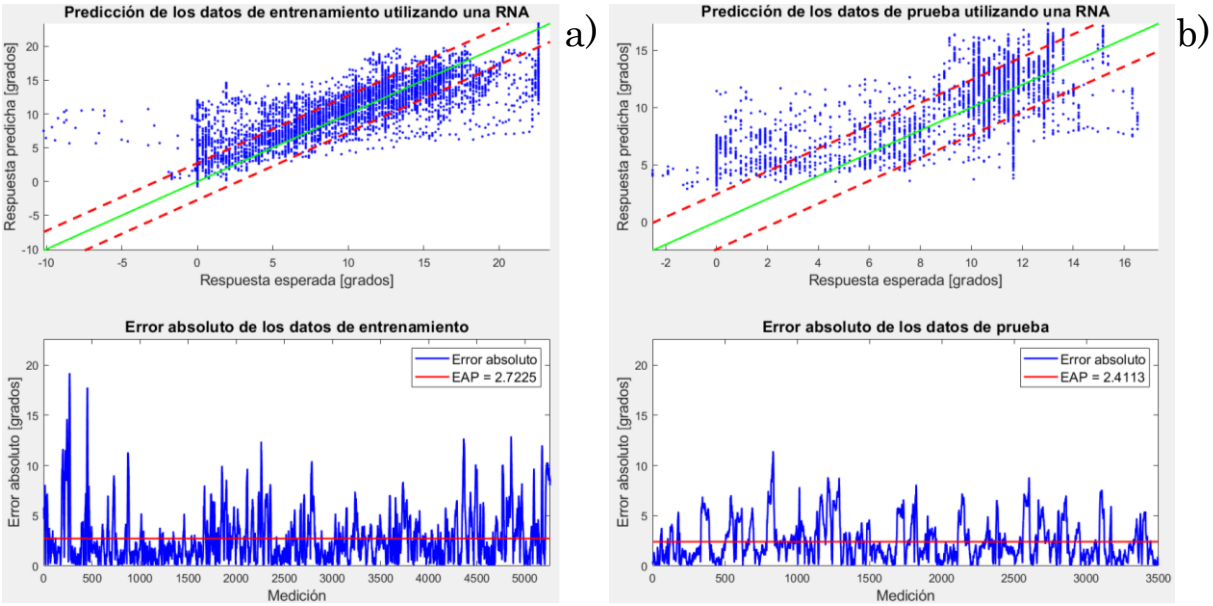


Figura 49. Comparación entre la respuesta predicha contra la esperada para un conjunto de datos de entrenamiento y prueba, en la parte inferior se muestra el error absoluto de estas.

De las figuras anteriores se puede observar que el modelo no se ajusta correctamente a la curva deseada, ya que en algunas regiones se aleja demasiado,

esto sucede principalmente en los extremos, lo que se puede deber a que se tiene pocos datos con estos valores.

Para tener una mejor visualización se graficó un histograma de los errores obtenidos por el modelo, con el fin de ver la dispersión de estos. Estos resultados se muestran en la Figura 50.

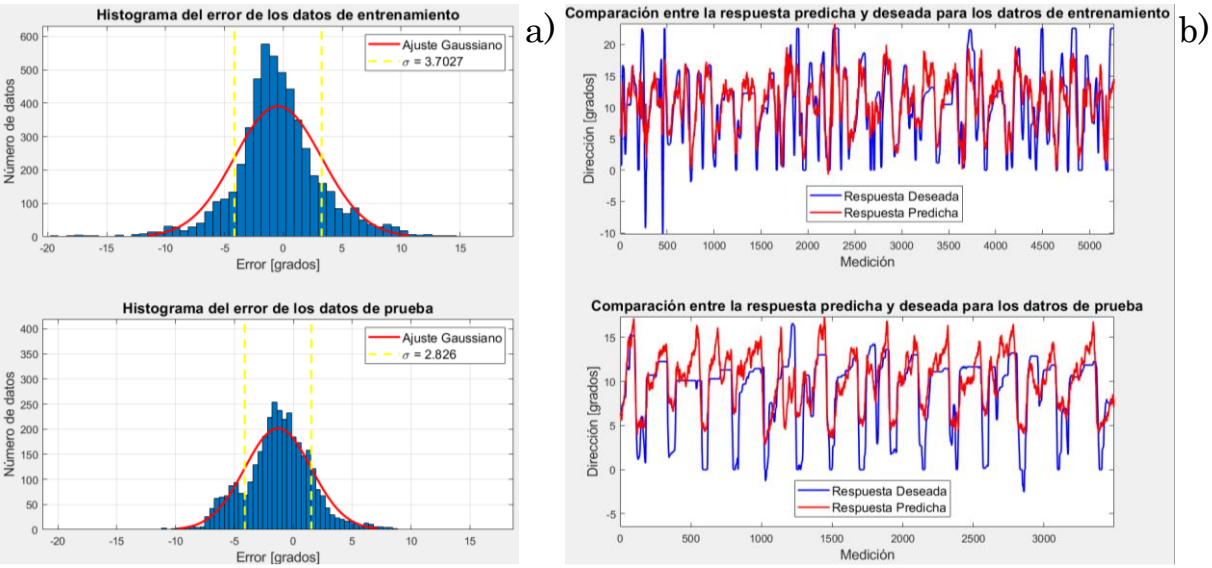


Figura 50 Histograma del error obtenido con la RNA para los conjuntos de entrenamiento y prueba, en b) se compara la respuesta deseada y la predicha para estos conjuntos.

Se puede observar que la desviación estándar de los datos de entrenamiento es menor que la obtenida con los datos de prueba, lo que confirma la hipótesis de que el modelo generado con esta red esta sobreajustado. Por otra parte, se puede ver en la Figura 50b que la predicción para los datos de entrenamiento y prueba siguen la tendencia general de los datos esperados, sin embargo, aun parecen contener señales ruidosas.

El sobreajuste obtenido con esta red neuronal se debe principalmente a un mal ajuste de hiperparámetros, por lo que, para obtener mejores predicciones, es necesario un mejor ajuste de estos.

En la siguiente tabla se mostrarán los resultados obtenidos con los tres métodos utilizando el conjunto 4 para el entrenamiento y el conjunto 5 como prueba, en esta tabla se muestra el error absoluto promedio (EAP) y la desviación estándar (STD) para los conjuntos de entrenamiento (Ent.) y prueba (Pru.).

Tabla 2. Resultados obtenidos con las tres técnicas estudiadas.

Técnica	EAP Ent.	STD Ent.	EAP Pru.	STD Pru.
PLS-R	**2.5333°**	**3.3015°**	**2.4413°**	**3.1571°**
PCR	2.5385°	3.32°	2.5378°	3.2586°
RNC	2.7225°	3.7027°	2.4113°	2.826°

Se puede observar que las técnicas de PCR y PLS-R tienen una buena generalización ya que el error absoluto promedio y la desviación estándar presentan un resultado similar en el entrenamiento y la prueba, por otra parte, se puede ver que el modelo producido por la red neuronal esta ligeramente sobre ajustado, se observó que al aumentar el número de épocas (más de 80), los resultados para el conjunto de validación siguen fluctuando y no disminuyen más de los resultados mostrados, por otra parte, estos resultados siguen disminuyendo para el conjunto de formación, este problema se puede solucionar utilizando validación cruzada durante el entrenamiento del modelo o ajustando adecuadamente los hiperparámetros.

Por otra parte, el tiempo para hacer predicciones mediante el uso de los modelos obtenidos con PLS-R y PCR son menores que el tiempo para hacer predicciones con la red neuronal, esto se debe a que PLS-R y PCR solo realizan un producto vectorial para hacer predicciones, mientras que la Red Neuronal de Convolución realiza más procesos para obtener una predicción. Por lo que se puede concluir que PLS-R y PCR son métodos más eficientes que la Red Neuronal, temporalmente hablando.

De la Tabla 2 se puede concluir que el mejor modelo generado es el obtenido mediante la técnica de PLS-R ya que tiene una buena generalización y presenta el menor error y desviación estándar para los conjuntos analizados.

Por último, para la visualización en tiempo real del error entre la dirección deseada y la predicha, se desarrolló una simulación que muestra la imagen analizada, obtenida del conjunto 5; para la visualización de la dirección se muestra un par de vectores que representan la dirección deseada y la predicha por el modelo y, por último, se muestra una gráfica que contiene el error de las últimas 100 mediciones.

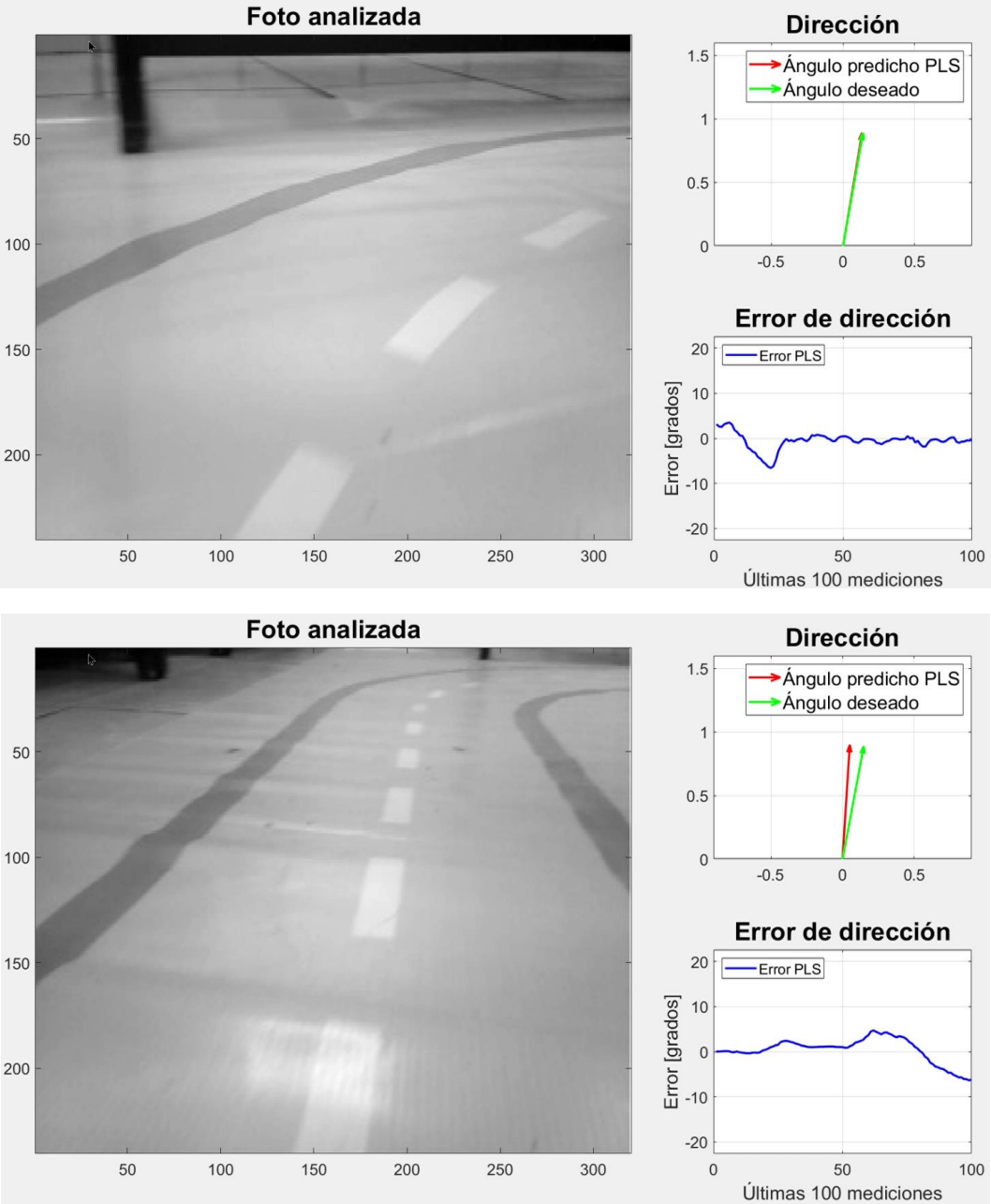


Figura 51. Simulación del funcionamiento del modelo generado con PLS-R, en esta se hace una predicción con respecto a la imagen analizada y se compara con la salida deseada para esta imagen, también se muestra el error de las ultimas 100 imágenes analizadas.

En la Figura 52 se muestra el error absoluto promedio y una comparación de la respuesta deseada contra la percha de los datos analizados con esta simulación.

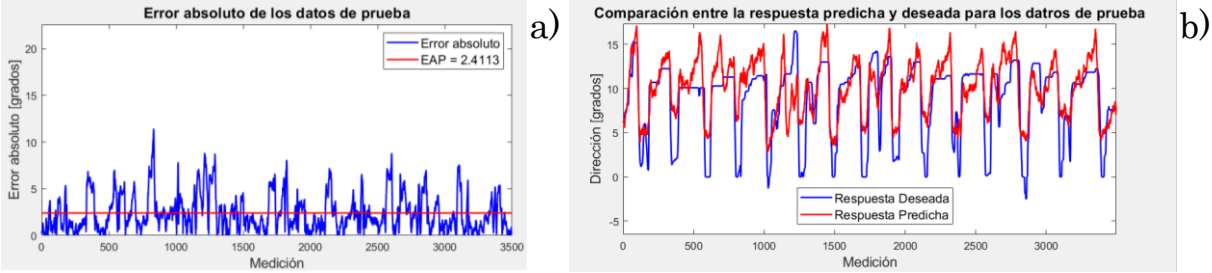


Figura 52. Error absoluto promedio de la predicción del ángulo y comparación de la respuesta deseada contra la respuesta predicha para el conjunto de datos analizados.

Los resultados obtenidos anteriormente con los distintos modelos se pueden mejorar obteniendo mejores datos de entrenamiento (conduciendo el prototipo correctamente), ocupando una computadora más potente como la Jetson Nano o la Mini PC MELE para la adquisición y análisis de datos y desarrollando un prototipo más complejo con más sensores para la adquisición de datos y control de este.

CONCLUSIONES

Después de analizar los modelos obtenidos para el manejo del prototipo, se llegó a las siguientes conclusiones:

I. La reducción de dimensiones de las imágenes analizadas no afecta significativamente al análisis de datos realizado mediante la técnica de PLS-R y PCR.

II. El mejor modelo producido mediante las técnicas analizadas fue obtenido ocupando el conjunto 4 para entrenamiento y el conjunto 5 como prueba.

III. El mejor modelo producido mediante el método de PLS-R fue el obtenido con cinco componentes y usando imágenes con una reducción de 240x320 pixeles a 60x80 pixeles, logrando así un error absoluto promedio del 2.44 ° y una desviación estándar del 3.15°, para un conjunto de prueba. Este resultado parecería ser bueno ya que estos valores son menores al 15% con respecto a un intervalo dinámico de 22.6°, sin embargo, si se quisiera implementar este modelo a una mayor escala este podría ocasionaría múltiples accidentes viales, por lo que el modelo generado con estos datos de entrenamiento no es el adecuado.

IV. El modelo generado mediante el uso de una red neuronal de convolución esta ligeramente sobreajustado, por lo que, para obtener mejores resultados, es necesario encontrar los hiperparámetros adecuados y obteniendo un mejor conjunto de datos de entrenamiento (entrenándolo de mejor manera).

V. Para reducir el tiempo de análisis de las imágenes tomadas por el prototipo, se deben optimizar los programas usados para la toma de datos, además de utilizar computadoras más potentes como lo son la Jetson Nano que contiene una tarjeta gráfica que se puede utilizar para el procesamiento de los programas o la Mini PC MELE que contiene un procesador más potente.

BIBLIOGRAFÍA

[1] Aurélien Géron. "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow". O'REILL, June 2019.

[2] Phil Kim, "MATLAB deep learning", Ed. Apress, 2017.

[3] Rafael C. González, Richard E. Woods, "Digital Image processing", Ed. Pearson Prentice Hall, 3rd Ed., 2008.

[4] Himanshu Singh, "Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python", Ed. Apress, 2019.

[5] Pérez Lima Dalí del Ángel, "Reconocimiento de patrones en imágenes mediante Aprendizaje Profundo", tesis de licenciatura 2021

[6] Chapman y Hall, "Introduction to self-driving vehicle technology", CRC press, 2020

[7] K. Mohaideen Abdul Kadhar, G. Anand. "Data Science with Raspberry Pi". Apress, 2021.

[8] Donald J. Norris. "Beginning Artificial Intelligence with the Raspberry Pi". Apress, 2017.

[9] Paul Bradt, Davis Bradt. "Science and Engineering Projects Using the Arduino and Raspberry Pi". Apress, 2020.

[10] Jeff Cicloani. "Beginning Robotics with Raspberry Pi and Arduino". Apress, 2021.

[11] Joseph Coburn. "Build Your Own Car Dashboard with a Raspberry Pi". Apress, 2020.

[12] Warren Gay. "Advanced Raspberry Pi". Apress, 2018.

[13] Guillermo Guillen. "Sensor Projects with Raspberry Pi". Apress, 2019.

[14] Donald J. Norris. "Machine Learning with the Raspberry Pi". Apress, 2020.

[15] https://www.adeept.com/video/static1/itemsfile/511343Tutorial.pdf

[16] https://www.adeept.com/video/static1/itemsfile/Introduction%20Motor%20HAT.pdf

[17] Svante Wold, Michael Sjöström, Lennart Eriksson, "PLS-regression: a basic tool of chemometrics", Elsevier, 2001

[18] https://scikit-learn.org/stable/auto_examples/cross_decomposition/plot_pcr_vs_pls.html

[19] https://es.wikipedia.org/wiki/Regresi%C3%B3n_de_m%C3%ADnimos_cuadrados_parciales

[20] Der Hau Lee. "Deep learning and control algorithms of direct perception for autonomous driving", Springer Nature, 2020.

[21] Chen C, Seff A, Kornhauser A, Xiao J "DeepDriving Learning affordance for Direct Perception in Autonomous Driving". Princeton University, 2015