

End-to-end learning para conducción autónoma

Jose Eduardo Laruta Espejo

8 de agosto de 2019

PyDay La Paz - Universidad Católica Boliviana

Contenido

1. Motivación
2. Conceptos básicos
3. Subsistema de Control y Actuación
4. Subsistema de Adquisición de Datos y Entrenamiento
5. Subsistema de Inferencia y control autónomo
6. Análisis de Resultados
7. Conclusiones y Recomendaciones

Motivación

¿Vehículos autónomos?



¿Vehículos autónomos?



¿Vehículos autónomos?



¿Vehículos autónomos?



¿Vehículos autónomos?

Un sistema de conducción autónoma cuenta con la siguiente arquitectura, en general:

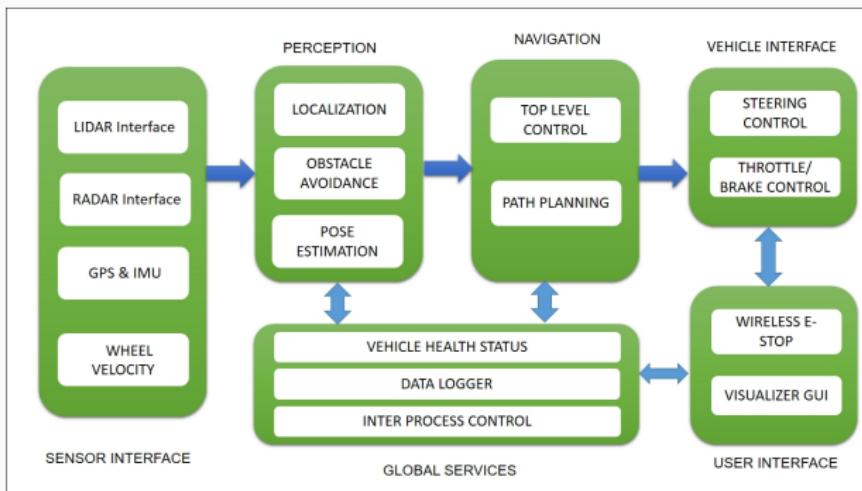


Figura 1: Esquema de un self driving car. Fuente: [1]

¿Vehículos autónomos?

Un **sistema de visión** para conducción autónoma cuenta con la siguiente arquitectura, en general:

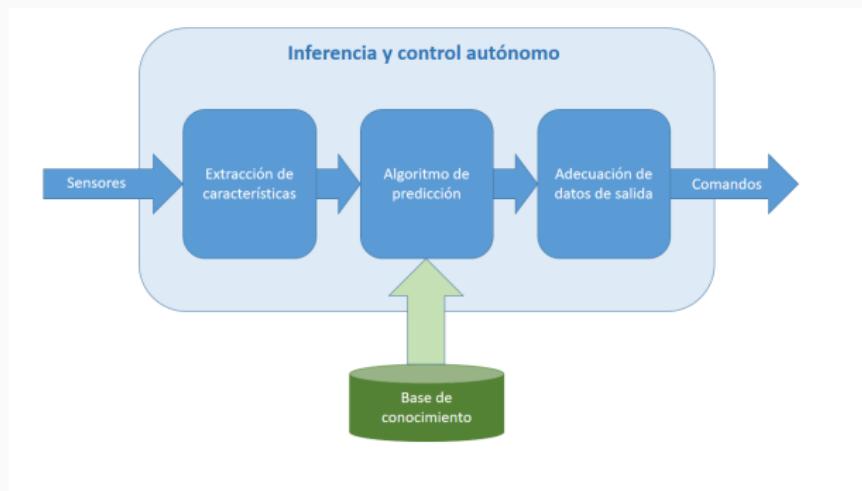


Figura 2: Componentes del subsistema de inferencia y control autónomo tradicional. Fuente: Elaboración propia.

¿Vehículos autónomos?

El sistema presenta las siguientes dificultades:

- **Conocimiento experto** en cada etapa.
- **Poca flexibilidad** en el diseño y la implementación.
- **Tiempo y recursos** necesarios elevados.



Objetivo

Diseñar un sistema de aprendizaje “*end-to-end*” capaz de generar de comandos de control de vehículos domésticos basado en visión artificial y **redes neuronales convolucionales** para facilitar el diseño e implementación de sistemas de conducción autónoma.

Conceptos básicos

Sistemas de Conducción Autónoma

Un sistema de conducción autónoma tiene la capacidad de operar un vehículo de forma completamente autónoma.



Figura 3: Vehículos del *Grand Challenge* de 2005: (a) Stanley (1er lugar), (b) Sandstorm (2do lugar). Vehículos del *Urban Challenge* de 2007: (c) Boss (1er lugar) (d) Junior (2do lugar). Fuente: [2]

Niveles de Autonomía SAE

Se han definido distintos niveles de autonomía por parte de la Sociedad de Ingenieros de Automoción (SAE).

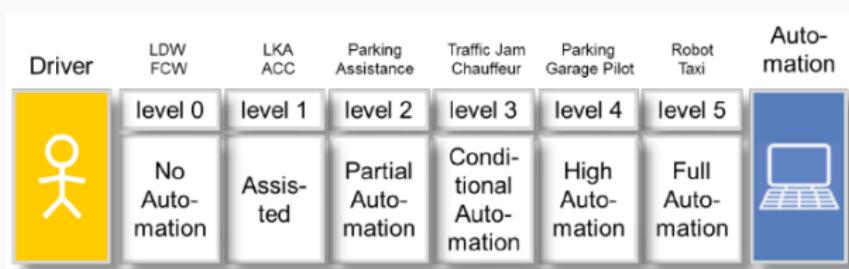


Figura 4: Niveles de automatización en la conducción según SAE. Fuente: researchgate

Visión Artificial

Es un campo de las ciencias de la computación que intenta reproducir las capacidades de una persona para entender imágenes. Aplicaciones:

- Reconocimiento óptico de caracteres (OCR).
- Inspección automática.
- Ventas.
- Reconstrucción de modelos en 3D (fotogrametría).
- Imagenología médica.
- Seguridad automotriz.
- Seguridad y vigilancia.

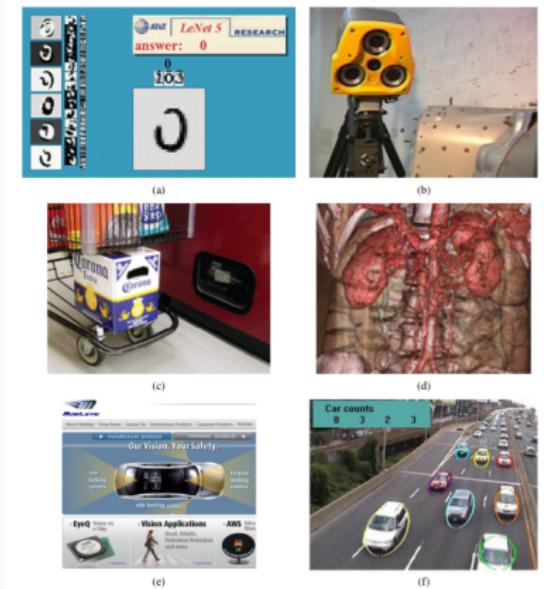


Figura 5: Algunas aplicaciones de la visión artificial. Fuente: [3]

Redes Neuronales Artificiales

Las redes neuronales artificiales son un modelo matemático de cómputo muy útiles en la aproximación de funciones y reconocimiento de patrones.

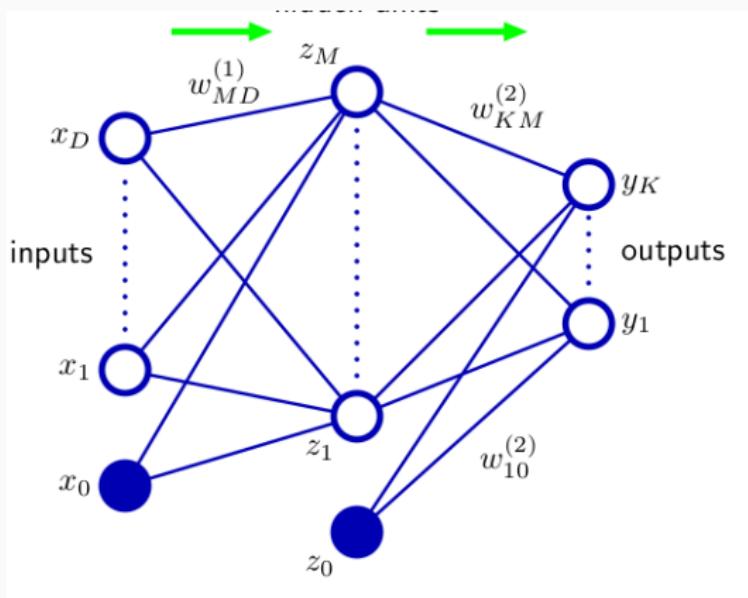
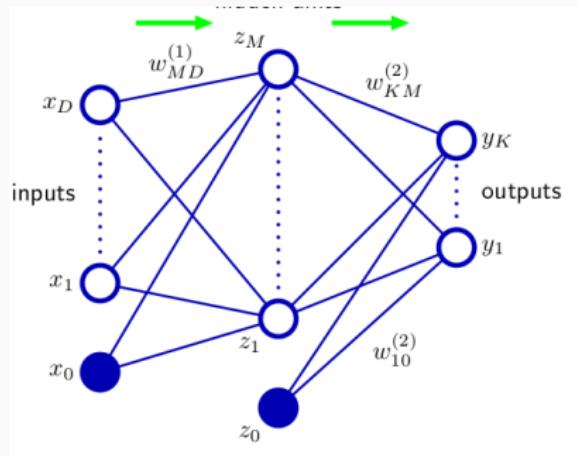


Figura 6: Diagrama de la red neuronal de dos capas. Fuente: [4]

Redes Neuronales Artificiales



Ecuación correspondiente:

$$y_k(\mathbf{x}, \mathbf{W}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \quad (1)$$

Función de Costo

La función de costo permite definir el rendimiento de la red con respecto a las predicciones esperadas a la salida.

Dado un conjunto de entrenamiento compuesto por un conjunto de vectores de entrada $\{\mathbf{x}_n\}$, donde $n = 1, \dots, N$, junto con un conjunto de vectores objetivo $\{\mathbf{t}_n\}$ el objetivo es **minimizar la función**:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 \quad (2)$$

Descenso de Gradiente

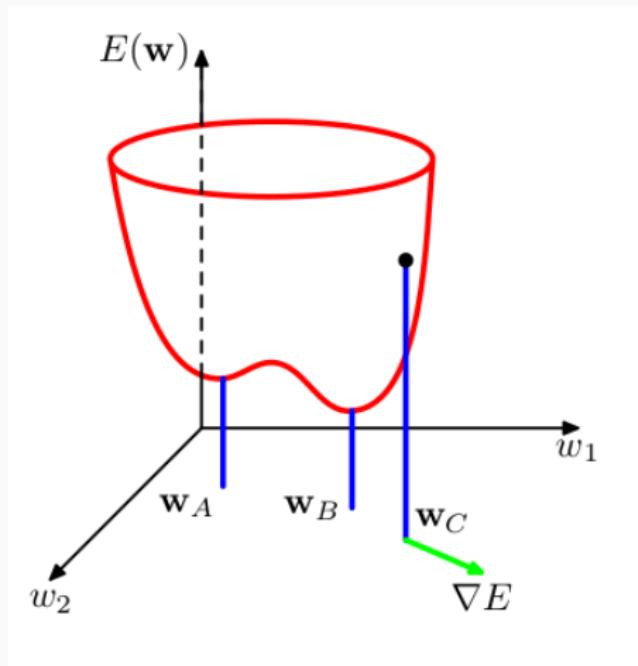


Figura 7: Visualización de la función de error $E(\mathbf{w})$ como una superficie. El punto \mathbf{w}_A es un mínimo local y el punto \mathbf{w}_B es el mínimo global. Fuente: [4]

Redes Neuronales Convolucionales

Las redes convolucionales usan la función de convolución para procesar la información en cada capa oculta.

La operación de convolución para señales digitales se define de la siguiente manera:

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t - \tau) \quad (3)$$

Para el caso de una imagen en 2D, la convolución se define como:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \quad (4)$$

Redes Neuronales Convolucionales

La operación de convolución se puede aplicar a imágenes (matrices multidimensionales):

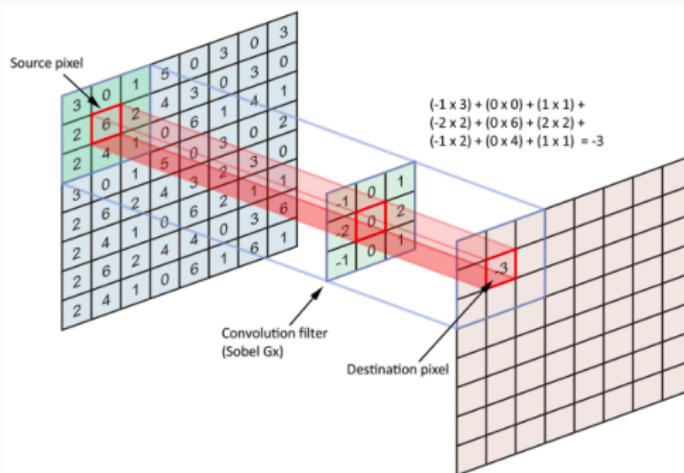


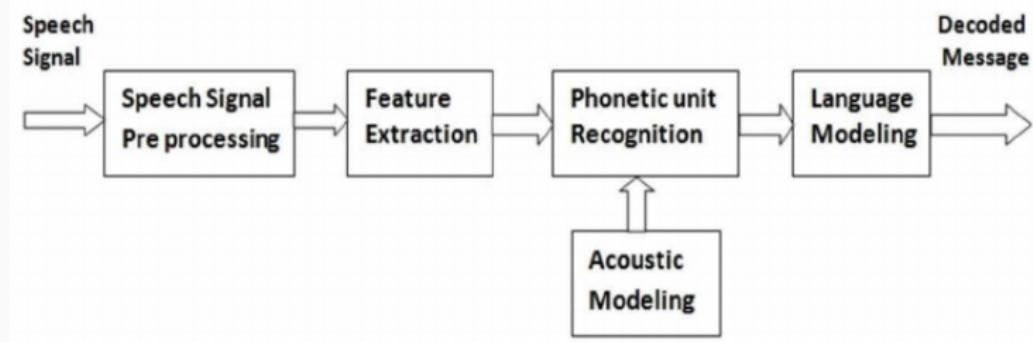
Figura 8: Visualización de la operación de convolución sobre una imagen digital. Fuente: [5]

Los sistemas de aprendizaje extremo a extremo son modelos de aprendizaje que tratan ciertos problemas **de manera conjunta** en lugar de separarlos en módulos o etapas de procesamiento.

Las representaciones internas de un sistema extremo a extremo se encargan de abstraer el conocimiento en base a la experiencia previa presentada en el conjunto de datos y no necesita de un experto que atienda el desarrollo de las mismas.

End-to-end learning

Un ejemplo de un sistema de aprendizaje tradicional (Reconocimiento de voz):

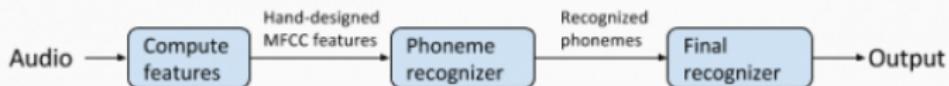


End-to-end learning

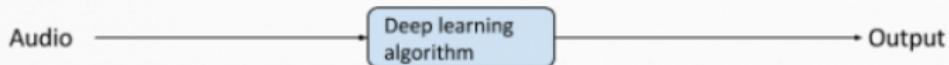
Un ejemplo de un sistema de aprendizaje extremo a extremo
(Reconocimiento de voz):

Speech recognition

Traditional model:



End-to-end learning:



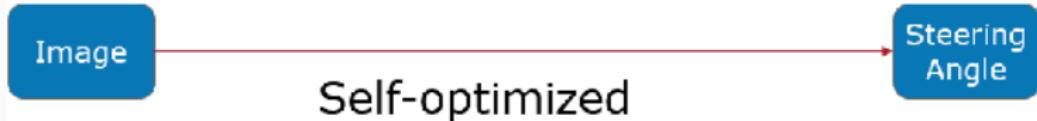
End-to-end learning

Para nuestra tarea:

Traditional approach

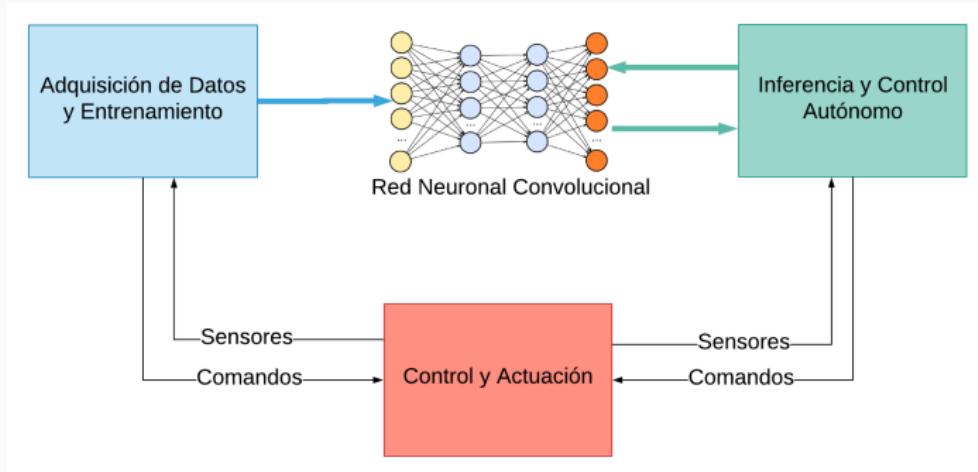


End to end learning



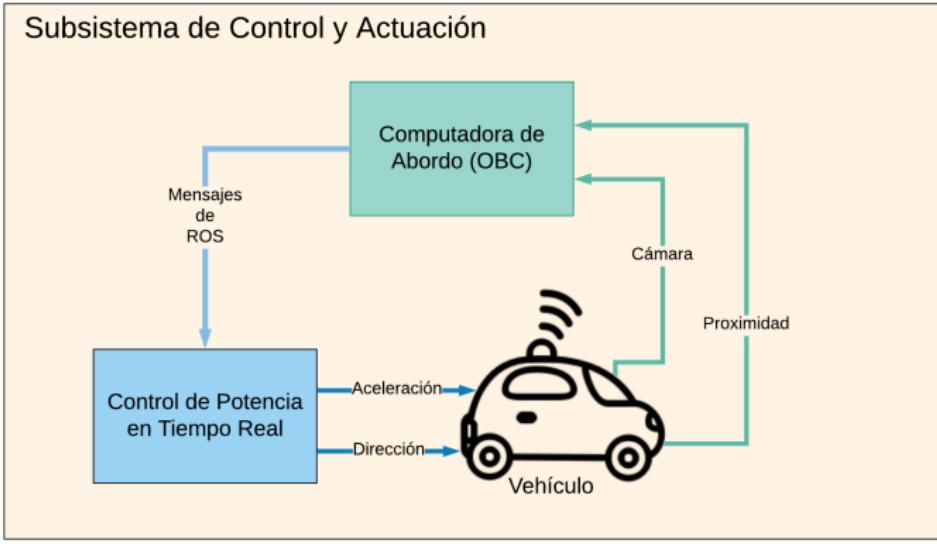
Ingeniería del proyecto

Arquitectura del Sistema



Subsistema de Control y Actuación

Esquema



Módulo de potencia en tiempo real

Se encarga del control de los actuadores mediante un sistema embebido que tiene capacidades de tiempo real¹.

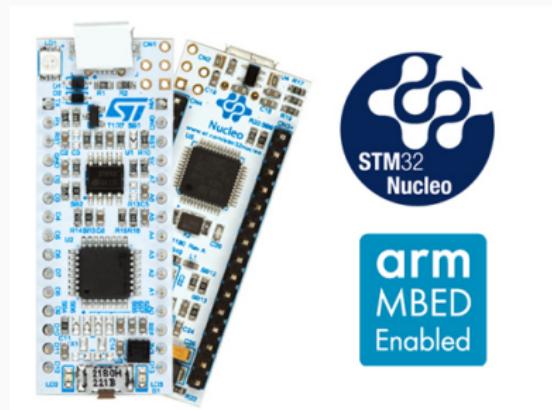
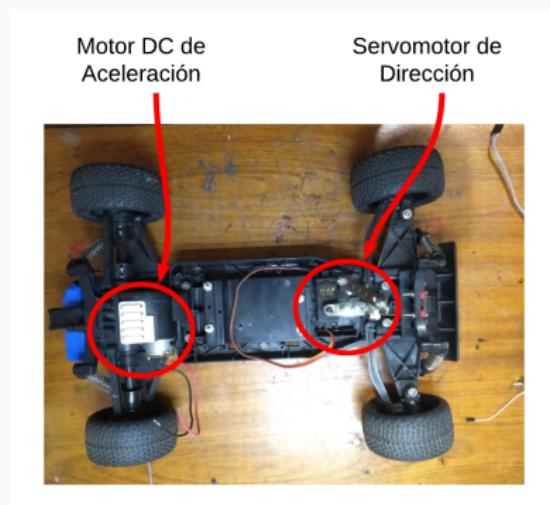


Figura 9: Placa de desarrollo NucleoF303k8 de ST Microelectronics.
Fuente: [6]

¹ Requerimientos de temporización estrictos. Tiempo de muestreo para el motor: 500 us

Módulo de la computadora de abordo (OBC)

La OBC se encarga del control de alto nivel del vehículo. Sus principales tareas son:

- Recepción de comandos de control remoto.
- Comunicación de los nodos de procesamiento del sistema.
- Envío de comandos de control al módulo de tiempo real.

Módulo de la computadora de abordo (OBC)



Figura 10: Fotografía de la OBC. Fuente: Elaboración propia.

Módulo de la computadora de abordo (OBC)

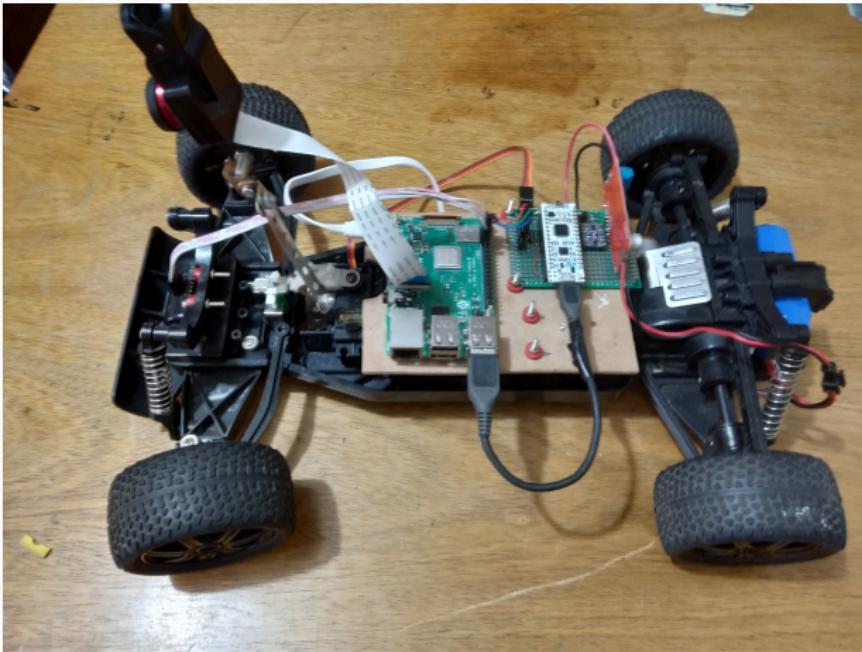


Figura 11: Fotografía del prototipo. Fuente: Elaboración propia.

Módulo de la computadora de abordo (OBC)

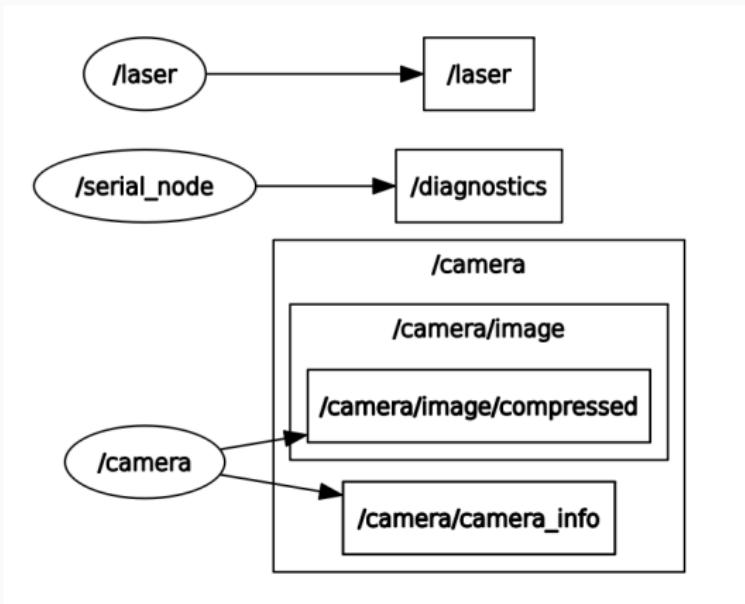
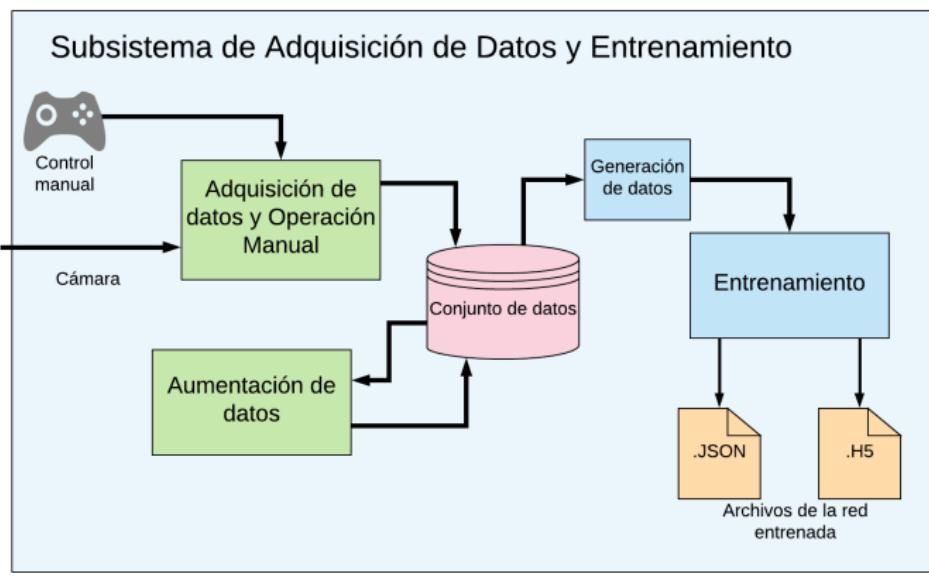


Figura 12: Interacción de los nodos en la OBC. Fuente: Elaboración propia.

Subsistema de Adquisición de Datos y Entrenamiento

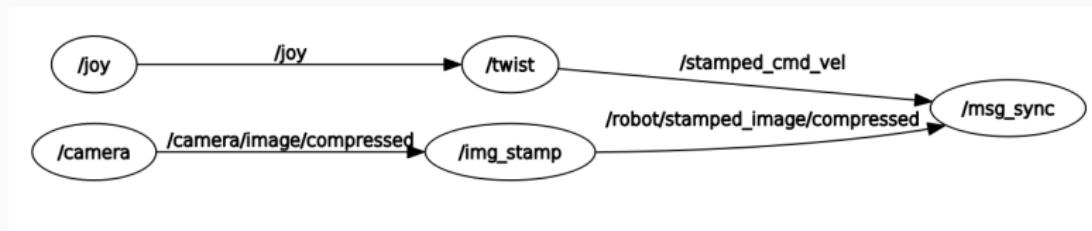
Esquema



Módulo de Adquisición de Datos y Operación Manual

Se realiza la adquisición de datos mientras se opera el vehículo manualmente.

Este modo de funcionamiento permite almacenar datos para el entrenamiento.



Módulo de Adquisición de Datos y Operación Manual



Figura 13: Imágenes capturadas por la cámara en una sesión de entrenamiento. Fuente: Elaboración propia.

Módulo de Aumentación de Datos

Una de las desventajas de las redes neuronales es que se necesita una cantidad sustancialmente superior de datos que en otros algoritmos de visión artificial.

Por su parte, la obtención del conjunto de datos es uno de los procesos más costosos en tiempo y recursos en un sistema de aprendizaje.

Módulo de Aumentación de Datos

Transformación	Descripción
Rotación	Rotación de la imagen en un ángulo aleatorio entre -20 y 20 grados.
Desplazamiento Vertical	Desplazamiento de la imagen de forma vertical un valor aleatorio entre 0 y 20 %.
Espejo Horizontal	Espejado de la imagen horizontalmente de forma aleatoria.

Tabla 1: Transformaciones realizadas en la aumentación de datos. Fuente:
Elaboración propia.

Módulo de Aumentación de Datos

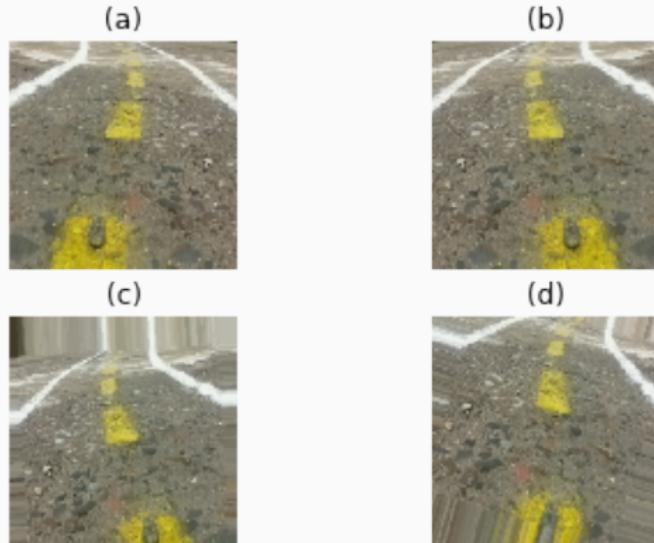


Figura 14: Ejemplos del proceso de aumentación de datos: (a) imagen original, imagen espejada horizontalmente, (c) imagen desplazada verticalmente, (d) imagen rotada. Fuente: Elaboración propia.

Módulo de Entrenamiento

Este módulo realiza las siguientes tareas:

- Cargar el dataset y el modelo de la red.
- Definir hiperparámetros.
- Ejecutar y monitorear el entrenamiento.
- Salvaguardar parámetros.
- Generar reportes del entrenamiento.

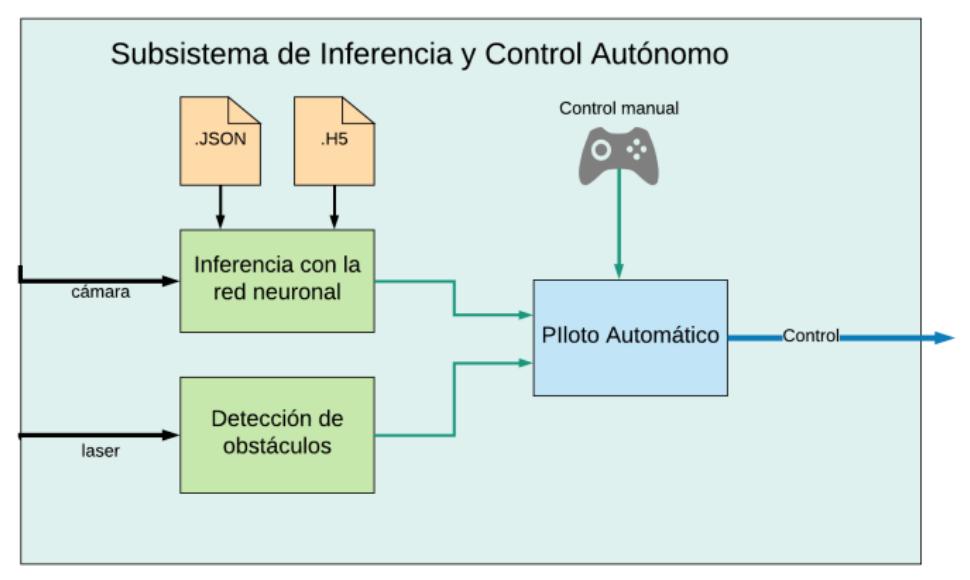
Módulo de Entrenamiento

El módulo tiene como salida dos archivos:

- **Arquitectura de la red neuronal:** Se almacena la información acerca de la arquitectura de la red en un archivo con formato *JSON* donde se puede encontrar la información acerca de las dimensiones de las capas ocultas, cantidad de unidades por capa y dimensiones de cada unidad en la red.
- **Pesos entrenados de la red neuronal:** Se almacena también los valores de todos los parámetros o pesos de la red neuronal, resultado del entrenamiento. Estos valores están almacenados en un archivo con extensión *H5*.

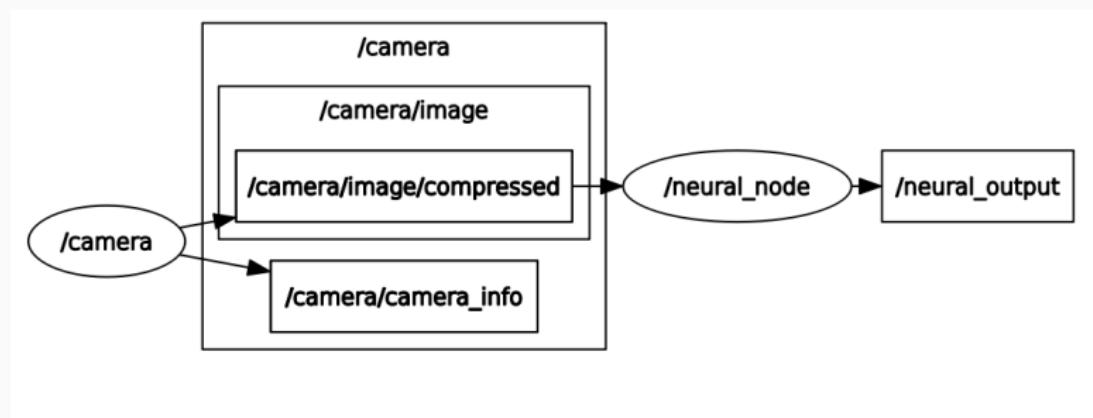
Subsistema de Inferencia y control autónomo

Esquema



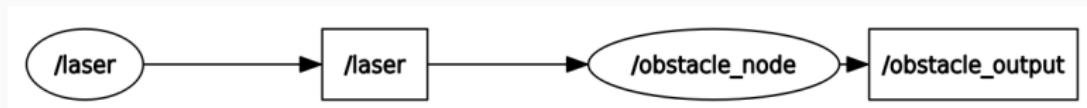
Módulo de Inferencia Neuronal

Este módulo tiene la tarea de ejecutar la tarea de predicción del comando de control de dirección con la red neuronal entrenada por el módulo de entrenamiento del subsistema de adquisición de datos y entrenamiento.



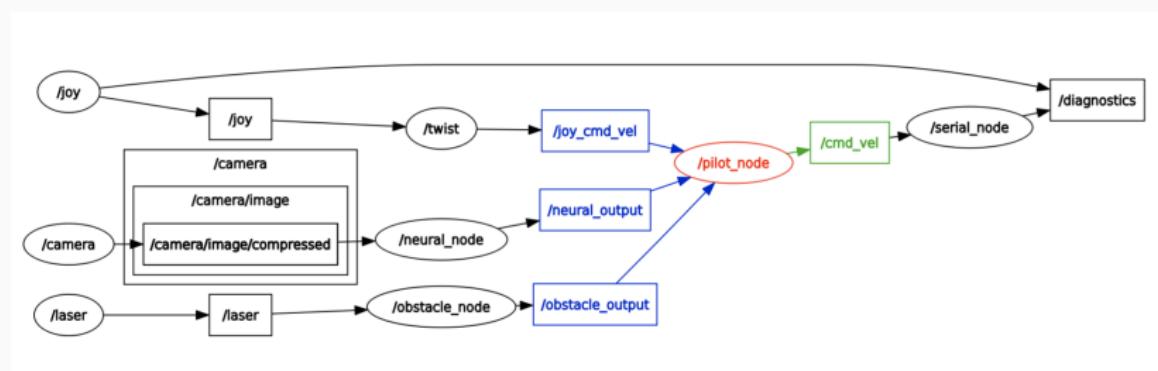
Módulo de detección de obstáculos

El módulo de detección de obstáculos tiene la finalidad de controlar la aceleración del vehículo basado en la presencia de obstáculos físicos frente al mismo. Se basa en las mediciones de proximidad realizadas por el sensor de proximidad montado en frente del vehículo.



Módulo del piloto automático

Este módulo se encarga de arbitrar los comandos de control provenientes de las distintas fuentes utilizadas en el sistema. Se suscribe a cada tema correspondiente y redirecciona los mensajes para el control del vehículo mediante mensajes.



Arquitectura de la Red Neuronal

Funciones de Activación - RELU

$$g(x) = \max\{0, x\} \quad (5)$$

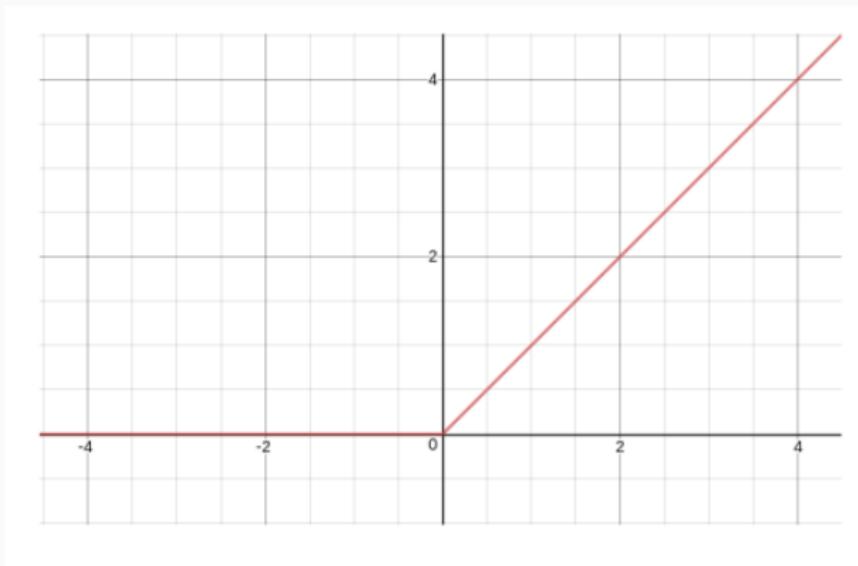


Figura 15: Gráfico de la función ReLU. Fuente: [7]

Funciones de Activación - Tanh

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (6)$$

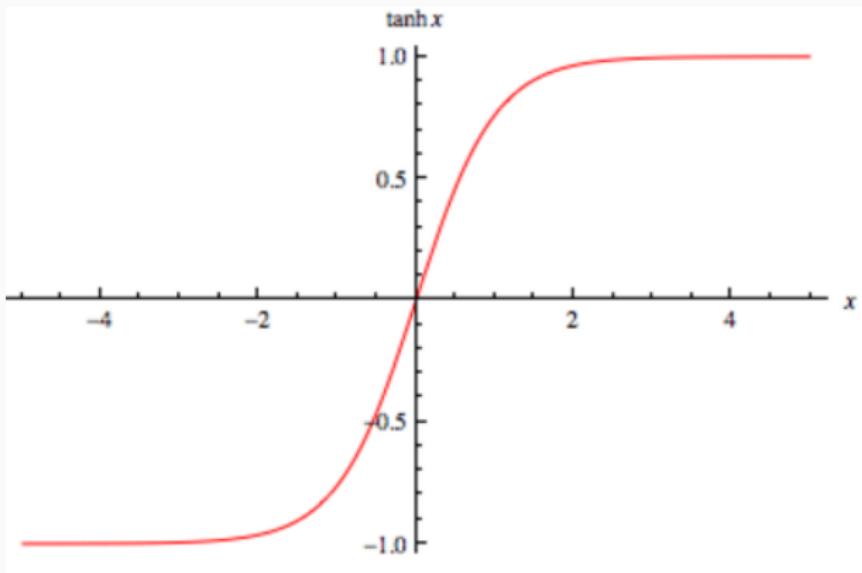


Figura 16: Gráfico de la función tangente hiperbólica $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
Fuente: [7]

Función de Costo

Se usa la función de **error cuadrático medio** como función de costo para la tarea de regresión, definida por:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 \quad (7)$$

Algoritmo de Optimización

Se ha utilizado el algoritmo **ADAM** [8] basado en momentos de primer y segundo orden.

$$\begin{aligned} m_w^{(\tau+1)} &= \beta_1 m_w^{(\tau)} + (1 - \beta_1) \nabla_w E^{(\tau)} \\ v_w^{(\tau+1)} &= \beta_2 m_w^{(\tau)} + (1 - \beta_2) (\nabla_w E^{(\tau)})^2 \\ \hat{m}_w &= \frac{m_w^{(\tau+1)}}{1 - (\beta_1)^{(\tau+1)}} \\ \hat{v}_w &= \frac{v_w^{(\tau+1)}}{1 - (\beta_2)^{(\tau+1)}} \\ w^{(\tau+1)} &= w^{(\tau)} - \alpha \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon} \end{aligned} \tag{8}$$

Implementación de Arquitecturas

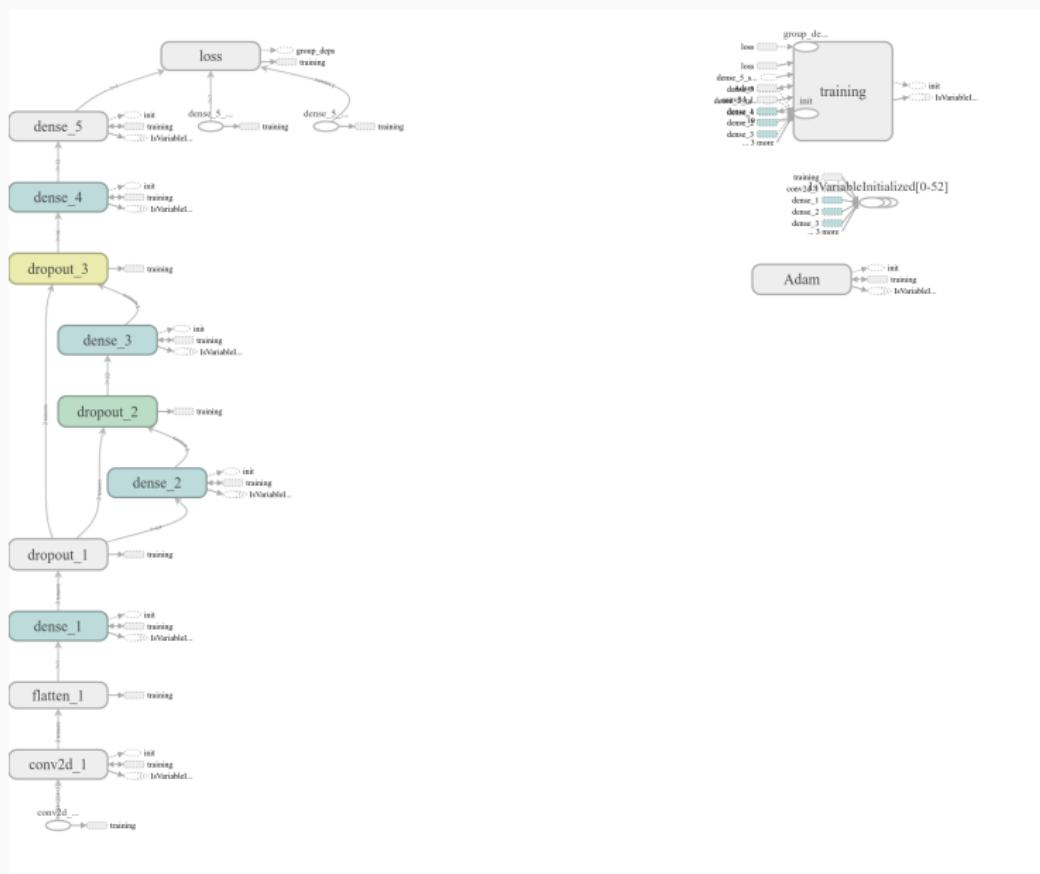
Se han implementado dos distintas arquitecturas de red neuronal con el propósito de comparar el rendimiento de una red neuronal convolucional comparado a una red tradicional.

Red neuronal tradicional

Capa	Tipo	Unidades (Filtros)	Dimensiones de salida	Parámetros entrenables	Función de Activación
input_1	InputLayer	-	(224,224,3)	0	
conv2d_1	Conv2D	3 (1 × 1)	(224,224,3)	12	ReLU
dense_1	Dense	64	64	9633856	ReLU
dropout_1	Dropout	-	64	0	-
dense_2	Dense	32	32	2080	ReLU
dropout_2	Dropout	-	32	0	-
dense_3	Dense	16	16	528	ReLU
dropout_3	Dropout	-	16	0	-
dense_4	Dense	12	12	204	ReLU
dense_5	Dense	1	1	13	Tanh

Total Capas	10
Total Parámetros	9636693

Red neuronal tradicional

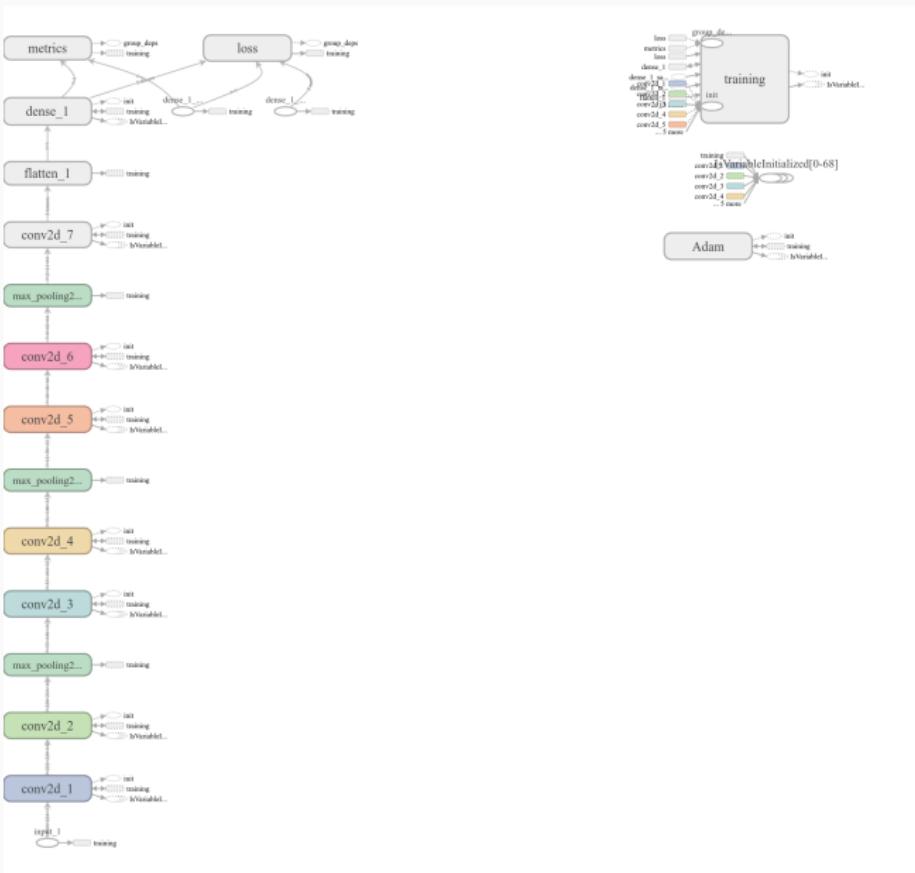


Red neuronal convolucional

Capa	Tipo	Unidades (Filtros)	Dimensiones de salida	Parámetros entrenables	Función de Activación
input_1	InputLayer	-	(224,224,3)	0	
conv2d_1	Conv2D	16 (3 × 5)	(222,220,16)	736	ReLU
conv2d_2	Conv2D	16 (3 × 5)	(220,216,16)	3856	ReLU
max_pooling2d_1	MaxPooling2D	-	(55,108,16)	-	-
conv2d_3	Conv2D	32 (3 × 5)	(53,104,32)	7712	ReLU
conv2d_4	Conv2D	32 (3 × 5)	(51,100,32)	15392	ReLU
max_pooling2d_2	MaxPooling2D	-	(12,50,32)	-	-
conv2d_5	Conv2D	64 (3 × 5)	(10,46,64)	30784	ReLU
conv2d_6	Conv2D	64 (3 × 5)	(8,42,64)	61504	ReLU
max_pooling2d_3	MaxPooling2D	-	(2,21,64)	-	-
conv2d_7	Conv2D	64 (3 × 5)	(2,21,4)	260	ReLU
dense_1	Dense	1	1	169	Tanh

Total Capas	12
Total Parámetros	120413

Red neuronal convolucional



Conjunto de datos

Balanceo del conjunto de datos

Debido a la naturaleza de las sesiones de entrenamiento, el conjunto de datos inicial se encuentra desbalanceado.

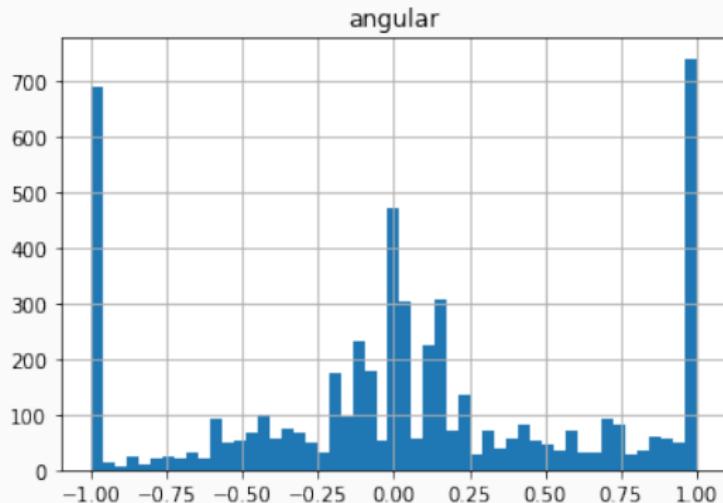


Figura 17: Distribución inicial del dataset. Fuente: Elaboración propia.

Balanceo del conjunto de datos

Se procede a realizar un balanceo del conjunto en base a dos tareas fundamentales:

- Eliminación de muestras con valor muy cercano o igual a cero.
- Aumentación de muestras con transformaciones aleatorias.

Balanceo del conjunto de datos

Una vez realizadas las operaciones se tiene un conjunto de datos con la siguiente distribución:

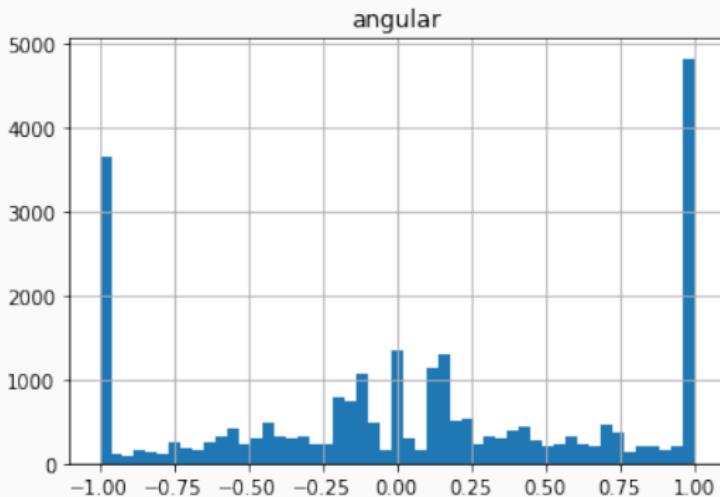


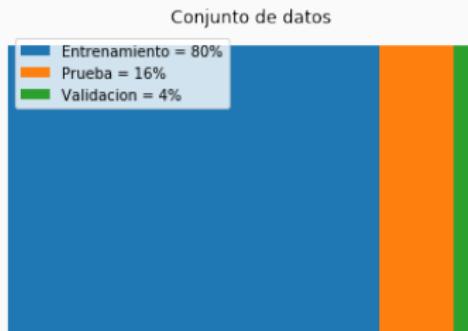
Figura 18: Distribución final del dataset. Fuente: Elaboración propia.

Separación de conjuntos de datos

Se ha separado el conjunto de datos en tres conjuntos con funciones distintas:

- **Entrenamiento:** Datos sobre los que se entrena la red neuronal en cada época.
- **Validación:** Pequeño conjunto sobre el cual se evalúa el error entre épocas.
- **Pruebas:** Conjunto no presente en el entrenamiento sobre el cual se realizan pruebas de rendimiento.

Conjunto de datos	Muestras	Fracción
Entrenamiento	21084	80 %
Prueba	3691	16 %
Validación	1581	4 %
Total	26356	100 %



Análisis de Resultados

Entrenamiento

Error de Entrenamiento

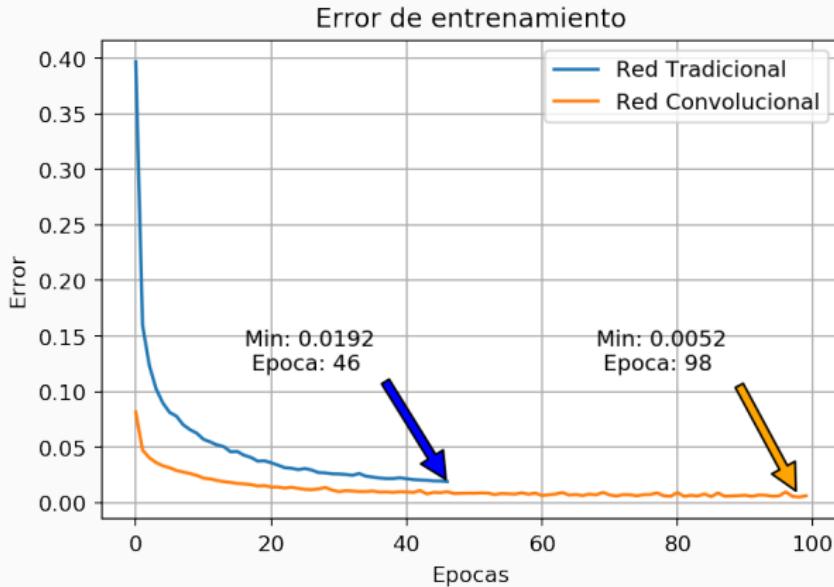


Figura 19: Error de entrenamiento. Fuente: Elaboración propia.

Error de Validación

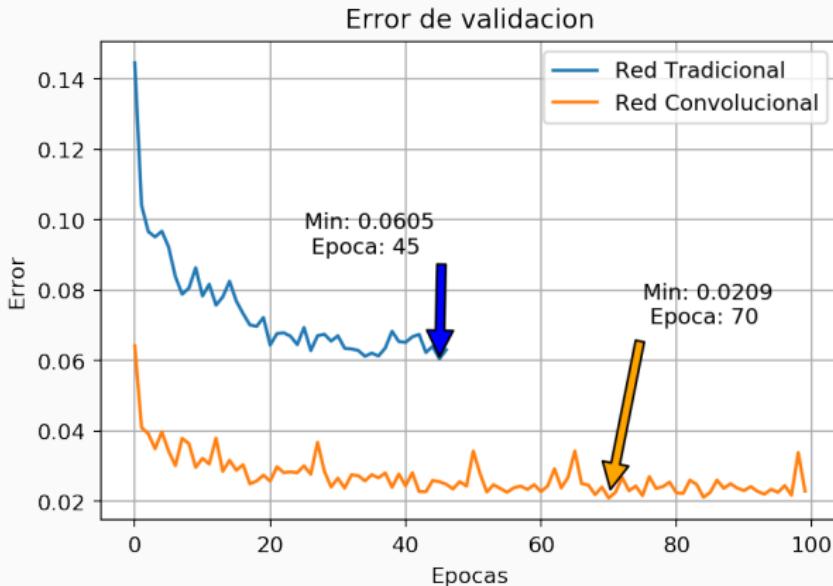


Figura 20: Error de validación. Fuente: Elaboración propia.

Pruebas

Puntajes y errores en el conjunto de prueba

Los puntajes sobre el conjunto de prueba definen el rendimiento y la capacidad de generalización de la red sobre datos nunca antes vistos. Se usan las siguientes métricas:

- **MSE**: Error Cuadrático Medio.
- **MAE**: Error Absoluto Medio.
- **R²**: Coeficiente de Determinación.

Modelo	MSE	MAE	R ²
Tradicional	0.0254	0.0976	0.9278
Convolucional	0.0160	0.0872	0.9484

Tabla 2: Evaluación de puntajes sobre el conjunto de prueba. Fuente: Elaboración propia.

Puntajes y errores en el conjunto de prueba

Valor real	Predictión	Diferencia	Predictión de signo correcta
-0.297	-0.243	0.054	✓
1.0	0.921	0.079	✓
0.149	0.22	0.07	✓
-1.	-0.968	0.032	✓
-0.	0.069	0.069	✗

Tabla 3: Muestra de predicciones de la red convolucional. Fuente: Elaboración propia.

Pruebas de Campo

Pred: [-0.536]



Pred: [0.760]



Pred: [-0.855]



Pred: [-0.088]



Pred: [-0.997]



Pred: [1.0]



Figura 21: Ejemplos de predicción de la red convolucional. Fuente: Elaboración propia.

Representaciones Internas



Figura 22: Filtros de la penúltima capa convolucional. Fuente: Elaboración propia.

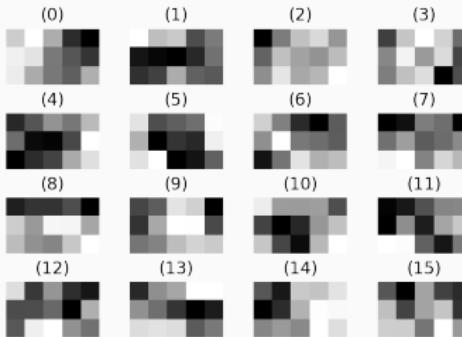
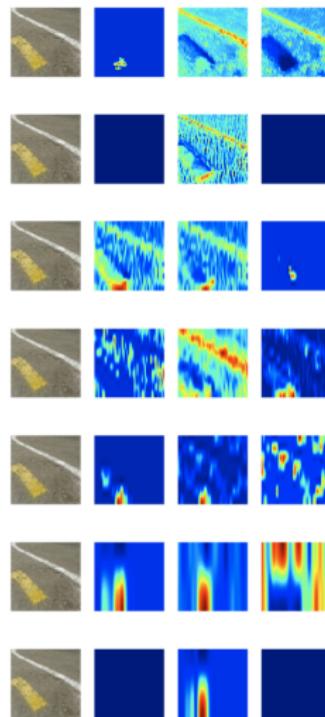


Figura 23: Filtros de la primera capa convolucional. Fuente: Elaboración propia.

Representaciones Internas

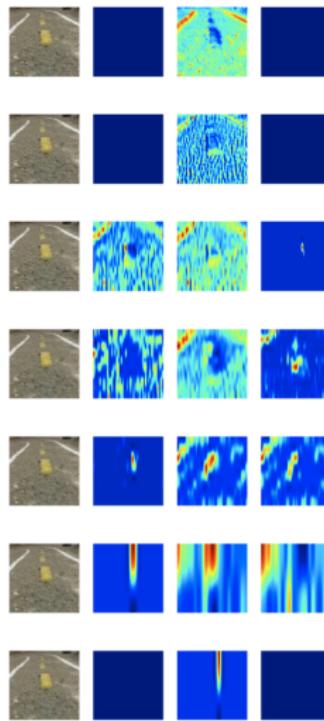
Giro a la izquierda.

Predicción: **1.**



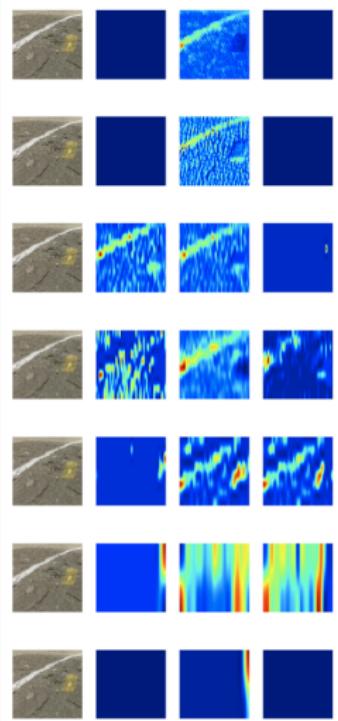
Hacia adelante.

Predicción: **-0.1.**



Giro a la derecha.

Predicción: **-0.84.**



Conclusiones y Recomendaciones

- Se estudiaron los aspectos relacionados al desarrollo de sistemas de conducción autónoma, y posteriormente los conceptos básicos de aprendizaje automático, aprendizaje profundo, redes convolucionales y el proceso de entrenamiento de una red neuronal.
- Se ha introducido el esquema de la arquitectura de un sistema de conducción autónomo y se han planteado los requisitos y funcionalidades que debe tener el sistema en su conjunto y por cada subsistema y módulos correspondientes.
- Se han diseñado todos los subsistemas correspondientes en base a las limitaciones y herramientas de software y hardware elegidas. Los subsistemas son:
 - Subsistema de Control y Actuación.
 - Subsistema de Adquisición de Datos y Entrenamiento.
 - Subsistema de Inferencia y Control Autónomo

Conclusiones ii

- Se ha implementado una red neuronal convolucional para la tarea de la conducción autónoma.
- Se ha entrenado de manera satisfactoria una red neuronal convolucional.
- Se ha validado el rendimiento de la red neuronal convolucional para tareas de visión artificial en base a una comparación con otra arquitectura tradicional.

Recomendaciones i

- Generar **distintos conjuntos** de entrenamiento en condiciones climáticas diversas y ambientes variados.
- Diseñar, entrenar y validar arquitecturas de **redes neuronales distintas** o con variaciones a una red neuronal convolucional tradicional.
- Extender la implementación de la red a una tarea de **clasificación categórica**.
- Implementar el sistema desarrollado en **otros modelos de vehículos**, como por ejemplo, tracción diferencial.
- Extender el sistema presentado en este proyecto con **tareas avanzadas**, incluyendo en el flujo de trabajo, módulos de planificación de trayectorias, detección de objetos y programación de misiones.

Recomendaciones ii

- Crear una **línea de investigación** dedicada al diseño e implementación de sistemas robóticos inteligentes y vehículos autónomos en el IEA.
- Considerar este proyecto y sus resultados como una **plataforma de desarrollo** sobre la cual se puedan implementar diversos algoritmos y aplicaciones en sistemas robóticos inteligentes y vehículos autónomos.

Preguntas?

Referencias i

-  L. Joseph, *ROS Robotics Projects: Make your robots see, sense, and interact with cool and engaging projects with Robotic Operating System.* Packt Publishing, 2017. [Online]. Available: <https://www.amazon.com/ROS-Robotics-Projects-interact-Operating/dp/1783554711?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1783554711>
-  “Linear regression,” Nov 2018. [Online]. Available: https://en.wikipedia.org/wiki/Linear_regression
-  R. Szeliski, *Computer Vision.* Springer-Verlag GmbH, 2011. [Online]. Available: https://www.ebook.de/de/product/9630757/richard_szeliski_computer_vision.html

Referencias ii

-  C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., 2006. [Online]. Available: https://www.ebook.de/de/product/5324937/christopher_m_bishop_pattern_recognition_and_machine_learning.html
-  D. Cornelisse, "An intuitive guide to convolutional neural networks," Apr 2018. [Online]. Available: <https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050>
-  "Nucleo-f303k8." [Online]. Available: <https://www.st.com/en/evaluation-tools/nucleo-f303k8.html>
-  T.-y. Wang, "From perceptron to deep learning," Jan 2016. [Online]. Available: <http://databeauty.com/blog/2018/01/16/From-Perceptron-to-Deep-Learning.html>

Referencias iii

-  D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.