EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Doctorado

Introducción de una capa de Agregación Neurodifusa para procesamiento de señales de dimensionalidad asimétrica

presentada por

M.C. Jesús Antonio Luna Álvarez

como requisito para la obtención del grado de

Doctor en Ciencias de la Computación

Director de tesis

Dr. Dante Mújica Vargas

Cuernavaca, Morelos, México. Septiembre de 2023.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Dedicatoria

A mis padres que son principal apoyo.
A mi esposa quien se quedó conmigo en todas las condiciones.
A mi hijo que es mi motivación para seguir.

Una vez más, agradezco con hechos . . .

Agradecimientos

Agradezco al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por el apoyo económico brindado durante mis estudios de doctorado. Al Tecnológico Nacional de México / Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET por brindar las instalaciones y permitirme realizar los estudios de doctorado.

De forma especial agradezco a el Dr. Dante Mújica Vargas quien más allá de dirigir mi investigación, ha sido un gran profesor, colega, compañero, pero sobre todo amigo. A él le atribuyo gran parte de quien soy ahora como investigador, profesionista, padre y persona.

A mi comité revisor, Dr. Manuel Mejía Lavalle†, Dr. Juan Gabriel González Serna, Dr. Noé Alejandro Castro Sánchez, Dr. Nimrod González Franco y al Dr. Jean Marie Vianney Kinani, por el tiempo invertido, sus aportaciones y comentarios para la mejora de esta investigación.

Agradezco a mis amigos del clan Nephilim, con quienes compartí tanto tiempo, conocimientos, trabajo, frustraciones, risas, comidas, hambres y grandes experiencias.

¡Gracias!

Resumen

La fusión o agregación de datos es una técnica de preprocesamiento que permite obtener información de distintas fuentes y homogeneizarla, de tal forma que el algoritmo de control sea provisto de información de mayor calidad. En esta investigación se propone la formulación e implementación de una técnica de Agregación Difusa expresada como una capa de una Red Neuronal de Aprendizaje Profundo. A diferencia de las propuestas existentes en la literatura donde se embebe el algoritmo dentro de la red o dentro de la función de activación, se propone formular una nueva neurona siguiendo los principios de parámetros ajustables w interpretados como Medidas Difusas μ autoajustables. La formulación de esta capa Neurodifusa implica la formulación de un algoritmo de ajuste adaptado a las medidas μ y compatible con los parámetros utilizados en las otras capas de un modelo neuronal profundo. El campo aplicativo de esta propuesta es en un sistema multisensorial, particularmente en la tarea de autoconducción de un vehículo. Se realizó experimentación en ambientes de simulación y en un prototipo a escala, obteniendo indices de autonomía superiores al 90%. Se realizó la comparación con métodos conocidos de la literatura para autoconducción y se demostró una mejoría del 9% con el enfoque propuesto, a la vez que con respecto a una conducción humana, se muestra una similitud del 98% con respecto a esta. El modelo de autoconducción propuesto ofrece una confianza de autonomía del 89% en el peor de los casos.

Palabras clave: Agregación Neurodifusa, Redes Neuronales Profundas, Medidas Difusas, Fusión de datos, Autoconducción.

Abstract

The fusion or aggregation of data is a pre-processing technique that allows obtaining information from different sources and homogenizing it, in such a way that the control algorithm is provided with higher quality information. This research proposes the formulation and implementation for Fuzzy Aggregation technique expressed as a layer in a Deep Learning Neural Network. Unlike existing proposals in the literature where the algorithm is embedded within the network or within the activation function, it is proposed to formulate a new neuron following the adjustable parameters w principles, interpreted as self-adjusting Fuzzy Measures μ. The Neurofuzzy layer formulation implies a training algorithm formulation, adapted to μ measures and compatible with the parameters used in the other deep neural model layers. The application field for this proposal is in a multisensory system, particularly in the task of self-driving a vehicle. Experimentation was carried out in simulation environments and on a scale prototype, obtaining autonomy rates of over 90%. The comparison was made with methods known from the literature for self-driving and an improvement of 9% was demonstrated with the proposed approach, while with respect to human driving, a similarity of 98% was shown with respect to this. The proposed self-driving model offers a range confidence of 89% in the worst case.

Keywords: Neurofuzzy Aggregation, Deep Neural Networks, Fuzzy Measurements, Data Fusion, Self-driving.

Índice General

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Este proyecto de investigación surge a partir de la necesidad de agilizar las cadenas de procesamiento en los sistemas multisensoriales equipados en los vehículos autoconducidos, como fue observado durante el desarrollo de la tesis de maestría [60]. En la Figura 1.1 se muestra la metodología empleada para el sistema de autoconducción antes mencionado, en el se observa que el bloque de fusión es paralelo a la función principal de control y codependiente del intercambio de información, agregando latencia al sistema y aumentando la complejidad computacional.

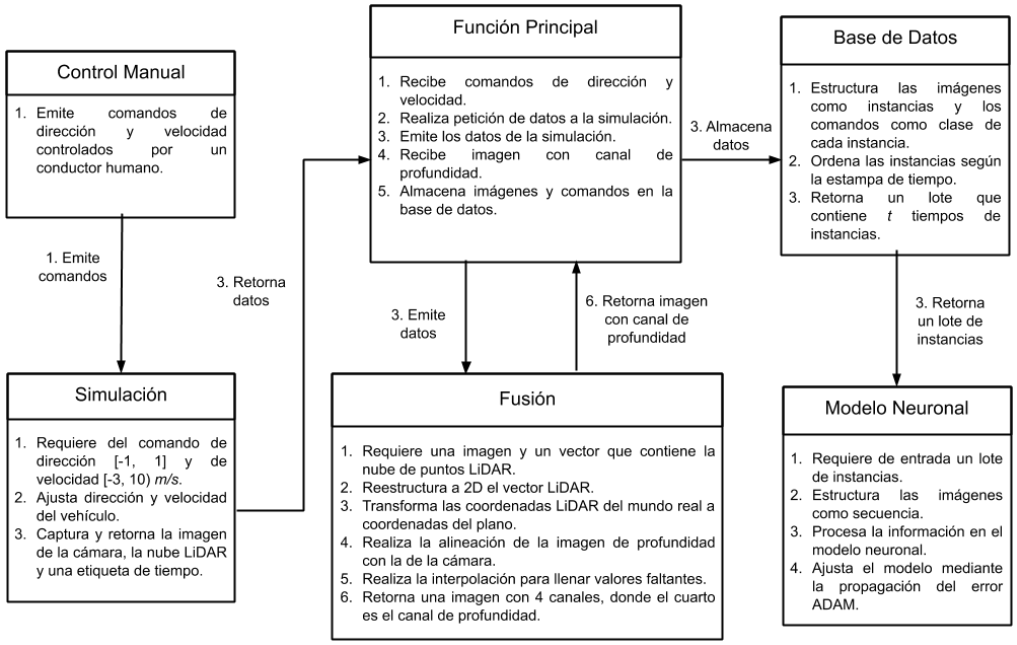


Figura 1.1 Metodología operacional del sistema de autoconducción [60]

En este aspecto, el sistema requiere de una arquitectura computacional de alto desempeño ya que debe realizar la obtención, limpieza, estructuración y fusión de datos, así como el procesamiento en el modelo neuronal para el control, lo que impide a ser implementado en un sistema embebido en el vehículo que es más adecuado para el espacio y consumo energético.

Esta área de oportunidad no se limita a la investigación de maestría, en la Figura 1.2 se muestra el modelo neuronal de detección de obstáculos [22] el cual muestra una arquitectura neuronal convolucional de procesamiento paralelo para cada fuente de datos y un bloque de fusión de información posterior a la extracción. En este sentido es equivalente a implementar tres modelo neuronales para detectar objetos. De forma similar otros trabajos emplean una metodología similar como es el caso de [21, 55, 28, 31], entre otros.

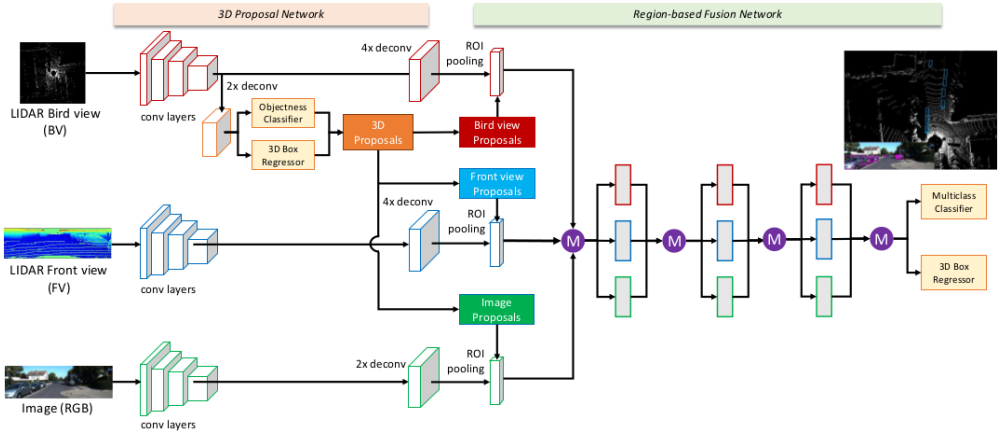


Figura 1.2 Red de detección de objetos 3D de vista múltiple [22].

Una solución a esta desventaja es la agregación de los sensores *a priori* al procesamiento de la Red Neuronal. La agregación es la compilación de información con la intención de preparar conjuntos de datos combinados para el procesamiento de datos [88]. En este sentido, se entiende la agregación de datos como un similar de fusión de datos. La fusión de datos puede realizarse a partir de distintos paradigmas, entre ellos:

- Métodos de estimación: es el método más simple y rápido que permite el procesamiento en tiempo real, basado en el uso de parámetros de ponderación sobre los datos redundantes, se define como una función de transformación $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Uno de los métodos predominantes es el Filtro de Kalman [51] ya que es capaz de identificar el estado oculto de un sistema y es tolerante a ruido aditivo [35].

- Métodos de inferencia: permiten inducir a partir de la información empírica proporcionada por una muestra, cual es el comportamiento de los conjuntos y su relación [97]. Entre los métodos más usados, la inferencia Bayesiana proporciona un formalismo para combinar conjuntos de acuerdo a las reglas de la teoría de probabilidades, representando la incertidumbre en términos de probabilidades condicionales [16].

- Algoritmos de Inteligencia Artificial: son métodos de inferencia de alto nivel, requieren de Reconocimiento de Patrones, deducción y aprendizaje que emulan

el razonamiento humano. Generalmente se basan en el uso de datos *a priori* y reglas de inferencia. Los paradigmas de Redes Neuronales y Lógica Difusa son ejemplos de estas técnicas [57].

En este sentido las Redes Neuronales de Aprendizaje Profundo son métodos eficaces para operar en sistemas multisensoriales. Sin embargo el paradigma original no tiene un método capaz de transformar el conjunto de fuentes en un conjunto combinado, sino que dependen del procesamiento de cada conjunto por separado, o bien de la fusión como un bloque de preprocesamiento. Por ello en esta investigación de pretende introducir al paradigma profundo un método de agregación combinando las propiedades neuronales con la lógica difusa a fin de eficientizar el procesamiento multifuente.

1.1 Planteamiento del problema

El problema que se desea resolver en esta investigación es el siguiente: Cómo darle a una Red Neuronal Profunda la capacidad de poder procesar en forma simultánea señales de diferentes fuentes y dimensionalidades. De tal forma que se pueda utilizar en aplicaciones más complejas como por ejemplo: vehículos autoconducidos, en donde se requiere procesar video y señales de sensores de proximidad de largo alcance; asistentes para personas invidentes, en los cuales se podría utilizar señales provistas por unidades de medición inercial, video y sobre todo señales acústicas.
Con base en la siguiente afirmación: “La Medida Difusa se puede obtener de varias maneras: definido por humanos, mediante operadores *Ordered Weighted Average* o bien mediante algoritmos de aprendizaje" [41], se plantea la siguiente hipótesis.

Hipótesis: la Agregación Difusa puede ser realizada por una capa neuronal cuyos parámetros w sean dados por Medidas Difusas y a su vez sean ajustados por un algoritmo basado en funciones objetivo, de tal forma que conserven la calidad de la fusión de información y sea realizada de forma autónoma.

1.1.1 Delimitación del problema

En esta investigación se desarrollarán las bases teóricas matemáticas y prácticas que definen a una capa neuronal como una capa de agregación basada en medidas difusas incluyendo la función de activación, así como el desarrollo teórico matemático y práctico del algoritmo de ajuste de parámetros para la capa propuesta. La implementación será evaluada y comparada en términos de calidad de fusión y clasificación con bases de datos propuestas en la literatura, y en términos operacionales aplicativos con el uso de sensores físicos.

1.1.2 Complejidad del problema

La complejidad de proponer una capa de agregación difusa en una red profunda consta de los siguientes puntos:

1. Al trabajar con señales $n-$dimensionales, la estructura de datos heterogénea requiere un procesamiento adaptable a la dimensionalidad y tamaño de la muestra.
2. La capa debe ser capaz de realizar la agregación sin necesidad de un preprocesamiento de los datos.
3. La capa debe ser compatible con otras capas de las redes profundas tales como las de convolución, recurrentes y densas.
4. El proceso de agregación no debe afectar negativamente a las tareas principales de clasificación, agrupamiento, control o cualquiera que realice principalmente el modelo de red neuronal.
5. Se debe diseñar un algoritmo de propagación del error especifico para ajustar los pesos sinápticos de la capa de agregación difusa.

1.2 Objetivos

1.2.1 Objetivo general

Proponer, formular y evaluar una capa difusa para redes neuronales profundas, la cual permita procesar señales de diferentes fuentes y dimensionalidades.

1.2.2 Objetivos específicos

- Analizar y formular matemáticamente las capas utilizadas en las Redes Neuronales de Aprendizaje Profundo.
- Analizar y formular matemáticamente medidas de agregación difusas aplicables al procesamiento de señales de dimensionalidad variable.
- Formular una capa de agregación difusa compatible con las Redes Neuronales de Aprendizaje Profundo.
- Formular una variante de algún método de entrenamiento para un modelo de Red Neuronal con capas difusas.
- Evaluar el modelo propuesto utilizando algún repositorio especializado de datos n-dimensionales.
- Comparar con otros métodos del estado del arte usando métricas especializadas.

1.3 Alcances y limitaciones

1.3.1 Alcances

- Estudiar los métodos de agregación difusa para fuentes de dimensionalidad variable.
- Estudiar los algoritmos de entrenamiento y procesamiento de las Redes Neuronales Profundas.
- Formular un método de agregación difusa interpretado como una capa neuronal con parámetros autoajustables.
- Formular un algoritmo de ajuste de parámetros basado en funciones objetivo para ajustar la capa difusa.
- Implementar el diseño, entrenamiento y prueba de los modelos neuronales en un lenguaje de bajo nivel paralelizado (GPU).
- Evaluar los resultados en términos de calidad de fusión, clasificación y operación del modelo neuronal mediante métricas especializadas del Estado del Arte.
- Comparar los resultados con distintas técnicas de fusión y clasificación, por separado y como un conjunto de procesamiento.

1.3.2 Limitaciones

- Para las fuentes de información se considerarán de 1 a 4 dimensiones.
- Se considera una dimensionalidad estática para cada fuente en entrenamiento y pruebas.
- La implementación será diseñada para procesar de 2 a 4 fuentes de datos o sensores simultáneos, siempre y cuando la profundidad y tamaño de los datos puedan ser procesados por las estructuras de datos y el *hardware*.
- Se espera que los datos estén adecuadamente estructurados.
- Debido a la escasa existencia de métricas especializadas en medir la calidad de la fusión y los *ground truth*, queda abierta la evaluación subjetiva de esta tarea.
- Se espera que la propuesta al menos iguale la calidad de fusión de otros métodos.
- La implementación se restringe a la capa difusa y aquellas que comprendan un modelo especializado, el algoritmo de ajuste formulado, lectura de datos y métricas seleccionadas. No se consideran otras capas neuronales existentes o algoritmos adicionales.

1.4 Justificación

La agregación o fusión de datos es un paso del preprocesamiento de datos que es utilizado en aplicaciones que requiere considerar más de una fuente de datos. Aunque existen múltiples técnicas que permiten fusionar los datos conservando la calidad, estos son manejados como un procesamiento extra previo a la tarea principal. Considerando la cualidad de las Redes Neuronales de ser autoadaptables, el proponer una capa de agregación basada en medidas difusas permite al usuario prescindir del preprocesamiento de los datos, además de ser compatible con la mayoría de las fuentes de datos.

1.4.1 Importancia

Con la elaboración de esta propuesta las principales aportaciones científicas que pueden obtenerse son las siguientes:

- Obtener el diseño de una capa de una Red de Aprendizaje Profundo que permita el procesamiento simultáneo de datos, con estructuras y dimensionalidad variable, obtenidos de varias fuentes, sin necesidad de realizar preprocesamiento.
- Proponer un algoritmo de entrenamiento para capas de agregación difusa basado en funciones objetivo.
- Proponer una capa neuronal entrenable basada en agregación difusa aplicable a tareas de Reconocimiento de Patrones con distintas fuentes de datos.
- Establecer las bases para el diseño de capas neuronales profundas extendido a otras aplicaciones como incremento de información, estructuración de datos, transformación de señales, compresión de datos, entre otras.
- Posicionarse en el Estado del Arte como un referente en el diseño de capas profundas para distintas aplicaciones.

1.5 Organización de la tesis

Este documento de tesis se compone por cinco capítulos principales, aunado a las secciones de anexos y referencias. El Capítulo 2 contiene el marco conceptual donde se describen los conceptos teóricos de estructuras tensoriales, necesarios para entender las redes neuronales de aprendizaje profundo, así como los conceptos de agregación difusa, necesarios para comprender la formulación de una estructura neurodifusa. En el Capítulo 3 se presenta el Estado del Arte, resumiendo los principales trabajos relacionados publicados en los recientes años, de la misma manera se exponen los antecedentes de tesis realizadas en la institución. En el Capítulo 4 se describe matemáticamente la teoría de agregación neurodifusa propuesta y su adaptación para la aplicación de autoconducción de vehículos. El Capítulo 5 detalla la experimentación realizada y los resultados obtenidos. Finalmente en el Capítulo 6 se describen las conclusiones generales de la investigación, trabajo futuro y los productos obtenidos.

Capítulo 2

Marco Teórico

En este Capítulo se proporcionan algunos conceptos importantes para el entendimiento de esta investigación, estos son desglosados en tres temas principales: (1) estructuras tensoriales que son la base estructural de las redes neuronales, (2) redes neuronales de aprendizaje profundo y los principales tipos de capas, donde se formaliza el algoritmo general para este paradigma, y (3) los conceptos de agregación difusa de datos.

2.1 Estructuras Tensoriales

Un tensor es una entidad algebraica que generaliza los conceptos de escalar, vector y matriz de una manera que sea independiente de cualquier sistema de coordenadas elegido [36]. Utilizando la notación indexada, el orden de un tensor será el número de índices necesario para especificar sin ambigüedad un componente de un tensor: un escalar será considerado como un tensor de orden 0; un vector, un tensor de orden 1 de cardinalidad n donde se expresa como $\sum_{i=1}^{n}$; y dada una base vectorial, los tensores de segundo orden pueden ser representados por una matriz de cardinalidad $m \times n$ donde se expresa como $\sum_{i=1}^{m}\sum_{j=1}^{n}$ [2].

El enfoque moderno visualiza los tensores inicialmente como objetos abstractos, construidos sobre espacios vectoriales abstractos, en los que se define un producto tensorial que permite construir estructuras típicas del álgebra multilineal [77]. Sus propiedades bien conocidas se pueden derivar de sus definiciones, como funciones lineales o incluso más generales; y las reglas para las manipulaciones de tensores se presentan como extensión del álgebra lineal al álgebra multilineal.

Álgebra multilineal

El álgebra multilineal es un área de estudio que generaliza los métodos del álgebra lineal. Los objetos de estudio son los productos tensoriales de espacios vectoriales y las transformaciones multilineales entre los espacios [13]. Para los tensores de orden uno,

usando la notación indexada: $X = X^s e_s$. Lo cual indica que el objeto X, es la combinación lineal:

$$\sum_{s=1}^{n} X^s e_s = X^1 e_1 + X^2 e_2 + \cdots + X^n e_n \tag{2.1}$$

sobre los vectores básicos e_s, y los X^s llamados los componentes de X. Aquí n es la dimensión algebraica de espacio donde reside X. Por convención se llama a estos contra-tensores.

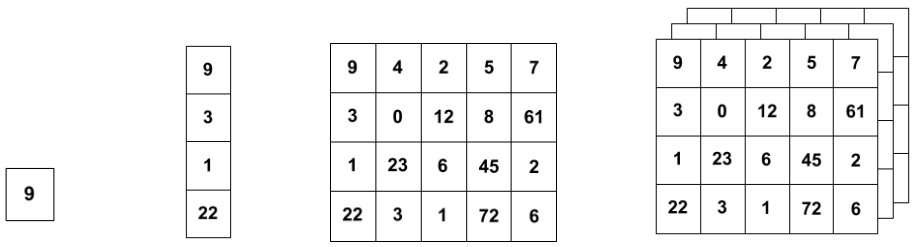
Un tensor de orden dos contravariante es $B = B^{st} e_s \otimes e_t$. Un tensor de orden dos covariante es $C = C_{st} e^s \otimes e^t$. Y un tensor de orden dos mixto es $D = D^s{}_t e_s \otimes e^t$. Esto indica una combinación lineal bi-indexada, si la dimensión del espacio es dos:

$$B = B^{11} e_1 \otimes e_1 + B^{12} e_1 \otimes e_2 + B^{21} e_2 \otimes e_1 + B^{22} e_2 \otimes e_2 \tag{2.2}$$

De forma general, un tensor es interpretado como un vector que contiene otros vectores. A partir de esta premisa es posible interpretarse como un arreglo multidimensional desde el punto de vista computacional.

Tensor computacional

Desde el punto de vista computacional, en un lenguaje de programación un tensor es definido como un arreglo mixto[73]. En el orden 0 se representa con una variable, el orden 1 de cardinalidad n es definido como un arreglo unidimensional de sintaxis $x[n]$. El orden 2 se interpreta como un arreglo de arreglos donde la primera dimensión contiene m arreglos que a la vez almacena n valores (reales). El orden 2 se interpreta como matriz, mientras el orden 3 se interpreta como un arreglo de matrices, es decir un arreglo de d arreglos que contiene m arreglos de n valores, en una sintaxis de lenguaje C se define un tensor 3D como $x[d][m][n]$ [52]. Esta lógica se interpreta de forma gráfica en la Figura 2.1.



(a) Real. (b) Vector. (c) Matriz. (d) Tensor 3D.

Figura 2.1 Ejemplo de estructuras tensoriales.

La Figura 2.1 es interpretada matemáticamente utilizando la notación indexada, de forma que para el vector de la Figura 2.1b se recorre mediante (2.3), la matriz 2.1c a partir de (2.4) y el tensor 2.1b mediante (2.5).

$$\sum_{i=1}^{n} x_i \tag{2.3}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \tag{2.4}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{d} x_{ijk} \tag{2.5}$$

Estos arreglos son definidos a partir de las características de la información que se procesa. Para el caso de una imagen, si se trata de escala de grices se considera un solo canal de color, por lo que un tensor 2D es definido a partir del alto y ancho de la imagen como se visualiza en la Figura 2.1c, sin embargo al procesar una imagen en el espacio de color RGB se consideran tres canales de color, por lo que la cardinalidad de $d = 3$, por lo que se visualiza como en la Figura 2.1d. En el caso de procesar un video se considera una dimensión adicional, el tiempo t en donde cada instancia es un cuadro o una imagen RGB contenida en un tensor 3D, de tal forma que el video se interpreta como un tensor 4D.

2.2 Redes Neuronales de Aprendizaje Profundo

El paradigma del Aprendizaje Profundo comprende de un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma tensorial [7]. El Aprendizaje Profundo tiene ventajas frente a otros algoritmos de aprendizaje automático gracias a sus tres características principales [25]:

- Robustez ante variaciones naturales en los datos, que permite a un modelo neuronal filtrar el ruido en los datos y así evitar las afectaciones al desempeño.
- Generalización que permite al un mismo modelo ser utilizado para otras aplicaciones o tipos de datos.
- Escalabilidad, puesto que el rendimiento puede mejorar con más datos de entrenamiento y de mayor calidad, dando oportunidad a un ajuste fino.

Estas características permiten que este paradigma sea aplicable en distintas tareas de Reconocimiento de Patrones como la clasificación, agrupamiento, así como detección

y reconocimiento de objetos en Visión por Computadora, Procesamiento de Lenguaje Natural, o bien control de sistemas y robots móviles. Se basa en algoritmos de Redes Neuronales derivados del Perceptrón el cual depende de parámetros ajustables *w* llamados pesos sinápticos, los cuales se adaptan a un conjunto de datos de entrenamiento con el uso de un algoritmo de propagación del error. Siguiendo este principio se plantean diversas capas compuestas por múltiples neuronas que realizan objetivos específicos.

Capas profundas

Considerando las estructuras de datos detalladas en la Sección 2.1, cada capa de una red profunda procesa los tensores a fin de realizar las tareas de normalización de datos, extracción de características y ajuste memoria.

Normalización

La capa de normalización propuesta por [5] aplica una transformación que mantiene la activación media de cada ejemplo cerca de 0 y la desviación estándar de activación cerca de 1 mediante la transformación:

$$\hat{X}_i = \frac{(X_i - \bar{X}_i)}{\sqrt{\sigma_i^2 + \varepsilon}} \cdot \gamma + \beta \tag{2.6}$$

donde $\bar{X}$ representa la media de cada entrada X, σ la varianza dentro de cada k características o atributos, ε es un hiperparámetro constante de estabilización con un valor de $1e^{-3}$, β y γ son parámetros ajustables de la capa.

La normalización por lotes *BN* [47] transforma una entrada $p-$dimensional $X = \{x_1^p, x_2^p, \cdots, x_n^p\}$ $BN_{\gamma,\beta} : x_i^p \mapsto \gamma \hat{x}_i^p + \beta$

$$BN_{\gamma,\beta}\left(x_i^p\right) \equiv \gamma \hat{x}_i^p + \beta \tag{2.7}$$

donde $\hat{x}_i^p = \dfrac{x_i^p - \mu_{\mathscr{B}}}{\sqrt{\sigma_{\mathscr{B}}^2 + \varepsilon}}$, $\mu_{\mathscr{B}} = \frac{1}{m}\sum_{j=1}^{m} x_j^p$ y $\sigma_{\mathscr{B}}^2 = \frac{1}{m}\sum_{j=1}^{m}\left(x_j^p - \mu_{\mathscr{B}}\right)^2$. Los parámetros γ y β son ajustados, ε es constante. En aplicaciones que implican procesamiento de imágenes, los canales de color RGB indican $p = 3$. En forma de notación, esta transformación se denomina como $\tilde{X} = BN_{\gamma,\beta}(x)$.

Extracción

La capa más utilizada en el Aprendizaje Profundo es la capa de Convolución [94], usada comúnmente para tareas de procesamiento de imágenes, información espacial, procesamiento texto, entre otras señales. La función principal es extraer características jerárquicas de bajo a alto nivel, que sean interpretables por otras capas o algoritmos a la vez que realizan el filtrado de anomalías en los datos con el uso de filtros ajustables w. La convolución puede ser realizada para señales de 1 a 3 dimensiones, con capacidad de expandirse a 4 dimensiones con el uso conjunto de capas recurrentes. Para la convolución 3D se opera a partir de k número de filtros que representan la profundidad de un tensor (canales RGB de una imagen a color), por lo que para obtener de salida k mapas de características se convoluciona:

$$\mathscr{C}_{i,j}^{k} = \sigma\left(\sum_{m=1}^{M}\sum_{N=1}^{N} X_{i+m,j+n}^{k} * W_{m,n}^{k} + b^{k}\right) \tag{2.8}$$

donde M y N representan el largo y ancho de una matriz dentro del tensor, W es el filtro ajustable y b el sesgo o *bias* relacionado a cada filtro. La convolución obtiene una expansión o reducción en profundidad k, sin embargo el tamaño de cada mapa de características se mantiene constante o decreciente en $m-2, n-2$, para realizar una reducción significativa y obtener solo aquellas características sobresalientes es necesario agregar una capa de agrupamiento, las cual puede operar a partir de un filtro de máxima (2.9) o de media (2.10) local.

$$\mathscr{P}_{i,j} = \max\left(\sum_{m=1}^{M}\sum_{n=1}^{N} y_{i+m,j+n}\right) \tag{2.9}$$

$$\mathscr{P}_{i,j} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} y_{i+m,j+n}}{M \times N} \tag{2.10}$$

En algunas aplicaciones es necesario expandir las características extraídas para obtener un tensor de salida similar al de entrada, tal como el ejemplo de segmentación del antecedente de [68]. El método utilizado para la expansión puede ser dado por el *upsampling* concatenando mapas de características, sin embargo un método más eficiente es la convolución transpuesta [32]. Dado el tamaño del mapa de características $X_w \times X_h$ y del filtro $W_w \times W_h$, esta capa obtienen mapas de características de tamaño $X_w + W_w - 1 \times X_w + W_w - 1$ como salida. Inicialmente el mapa de salida $\mathscr{C}^T$ se inicializa en cero con el tamaño esperado:

$$\sum_{i=0}^{X_w+k_w-1}\sum_{j=0}^{X_h+k_h-1}\mathscr{C}_{i,j}^{T} \leftarrow 0 \tag{2.11}$$

La convolución transpuesta para un solo mapa de características se define como:

$$\mathscr{C}_{i+m,j+n}^{T} = \sum_{i=0}^{X_w}\sum_{j=0}^{X_h}\sum_{m=0}^{k_w}\sum_{n=0}^{k_h} X_{i,j} * W_{m,n} \tag{2.12}$$

Por ejemplo, la convolución transpuesta de una entrada de 2×2 y kernel 2×2:

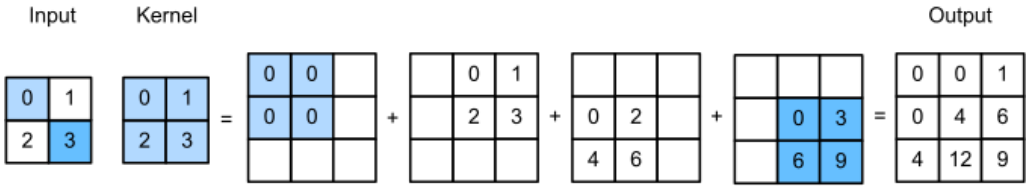


Figura 2.2 Convolución transpuesta con kernel 2×2.

Independientemente de la extracción realizada por la convolución, es necesaria una función de activación que transforme las características en valores escalares que ajusten las siguientes capas o realicen una clasificación, para ello y considerando **x** como la salida de la operación de cualquier capa, se emplean las funciones de activación.

Memoria

Las capas de memoria pueden ser dadas por neuronas Perceptrón Multicapa que opera a través del producto interno $\langle x \circ W \rangle$ y dando una salida oculta h por medio de una función de activación. Este tipo de capas son usadas para ajustar la red a partir de los datos obtenidos de las capas de extracción [43]. Para problemas más específicos de dependencia temporal se proponen las capas recurrentes, las cuales pueden ser capas *Long Short-Term Memory*, *Gated Recurrent Unit* [24] y demás variantes que siguien el mismo principio: el uso de compuertas neuronales dadas por parámetros W_r donde cada r representa una función especifica tal como el reinicio a 0 de un estado de memoria, de la misma manera cada neurona se autoalimenta con un tensor R que ajusta la neurona en el tiempo t mediante R_{t-1}. En la generalidad, una capa recurrente genera estados ocultos h_t activados por una función $\sigma(h_t)$, sin embargo no son capaces de realizar clasificación por sí solas, por lo que es necesario reducir la expansión del tiempo t en un tiempo final τ y generar una salida y a partir de:

$$y = \sigma_o[\tau] \circ \sigma(h_\tau) \tag{2.13}$$

donde σ_o representa la influencia de una compuerta de olvido autoajustada para determinar si la salida es relevante para el aprendizaje.

Otra aplicación de este tipo de capas es el reinicio de ciertas entradas aleatorias con el fin de evitar el sobreajuste de la red, por ello se implementan capas *Dropout* (2.14) [87], que bajo un parámetro aleatorio p escala la información obtenida en las capas anteriores teniendo la probabilidad de reducir a 0 el valor de alguna neurona, generalmente son usadas entre capas MLP.

$$\begin{aligned} \mathscr{D}_i &= \mathscr{R}3 * p_i \\ 0.2 &\leq p_i < 1.0 \end{aligned} \tag{2.14}$$

Funciones de activación

La función de activación más básica es la umbralización binaria (2.15) utilizada en el Perceptrón, sin embargo esta solo es aplicable en problemas de separación lineal. Para problemas de separación no lineal se plantea el uso de la función sigmoidal (2.30) para salidas en el intervalo $[0,1]$ y la función Tangente Hiperbólica (2.17) para salidas en el intervalo $[-1,1]$ con mayor restricción y tendencia a los extremos.

$$y = \begin{cases} 1, & \mathbf{x} \geq \theta \\ 0, & \mathbf{x} < \theta \end{cases} \tag{2.15}$$

$$\sigma(\mathbf{x}) = \frac{1}{1+e^{(-\mathbf{x})}} \tag{2.16}$$

$$\sigma(\mathbf{x}) = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}} \tag{2.17}$$

En el caso de las capas de convolución es necesario rectificar las salidas para evitar valores negativos que representan ruido en los mapas de características, para ello se utiliza la función Unidad Lineal Rectificada ReLU $\sigma(x) = \max(0,x)$. Para aplicaciones donde se requiere una rectificación que aproxime los valores a 0, la función Unidad Exponencial Lineal ELU es propuesta, donde α es una constante cercana a 1.

$$\sigma(\mathbf{x}) = \begin{cases} \mathbf{x}, & \mathbf{x} > 0 \\ \alpha(e^{\mathbf{x}} - 1), & \mathbf{x} \leq 0 \end{cases} \tag{2.18}$$

En las últimas capas de un modelo neuronal es necesaria una función de activación no lineal que realice la clasificación. Para aplicaciones específicas en un dominio continuo es posible utilizar la función Sigmoidal (2.30) como salida, sin embargo para aplicaciones

multiclase en el dominio discreto es necesario interpretar las salidas como la elección de una clase, para eso la función de activación *SoftMax* (2.19) [37] puede ser interpretada como una capa del modelo, la cual se comprende de un vector de probabilidades:

$$\sigma(\mathbf{x})_j = \frac{e^{\mathbf{x}_j}}{\sum_{i=1}^{M} e^{\mathbf{x}_i}}, j \in \{1,\ldots,M\} \tag{2.19}$$

donde M es el total de clases posibles. Para seleccionar la clase se usa una función $y = \arg\max[\sigma(x)]$.

Optimizadores y algoritmos de ajuste

El algoritmo de Retropropagación del error es la base del aprendizaje de las redes neuronales [40]. El algoritmo consta en el cálculo del gradiente en dos fases: la primera consiste en la propagación hacia adelante realizando lo explicado en la Subsección 2.2 para cada caso en las capas profundas hasta generar una salida. La segunda fase consta de comparar la salida obtenida y con una salida esperada y' o también llamada *ground truth*. La comparación consta del cálculo de una medida de distancia generalmente dada por el Error Cuadrático Medio (2.28).

$$E(W) = \sum_{i=1}^{I} ||\langle x_i \cdot W\rangle - y_i'||^2 \tag{2.20}$$

El método general para minimizar el error es el actualizar los parámetros de manera iterativa al sumar un incremento ΔW al valor actual: $W := W + \Delta W$. Por el contrario si se utiliza una función $\mathscr{N}(x,W)$ para aproximar los valores de salida y es diferenciable con respecto a W es posible usar como algoritmo de aprendizaje el método de Gradiente Descendente:

$$\Delta W = -\alpha \frac{\partial E(W)}{\partial W} \tag{2.21}$$

donde $0 < \alpha < 1$ es un parámetro conocido como tasa de aprendizaje. En un modelo completo, las capas ocultas son actualizadas mediante la derivada de la función de activación, por ejemplo las capas activadas por la función sigmoidal (2.30) son ajustadas mediante (2.31). El algoritmo se detiene cuando el error alcanza un mínimo valor deseado.

$$\frac{d\sigma(\mathbf{x})}{dx} = \sigma(\mathbf{x})(1-\sigma(\mathbf{x})) \tag{2.22}$$

En este punto es importante mencionar la diferencia entre iteraciones y épocas de entrenamiento: un conjunto X de ejemplos se divide en dos conjuntos destinados a entrenamiento y validación, para cada conjunto $x \in X$ consta de $|x| = I$ ejemplos, que a su

vez pueden permutarse en lotes de j ejemplos. El proceso de propagación hacia adelante y atrás en j veces es definido como una iteración, una época se conforma de k iteraciones de entrenamiento. El proceso de ajuste y validación es alternado entre épocas, así obteniendo resultados similares a la validación cruzada y permitiendo el ajuste posterior [67].

Optimización del error

Existen variaciones del algoritmo GD como el Gradiente Descendente Estocástico [12] que incluye la aleatoriedad en favor de encontrar el mínimo de una función, así como el Gradiente Descendente por Lotes [79] el cual calcula la media de los gradientes calculados en una época de entrenamiento. Aunque estas técnicas son ampliamente usadas para optimizar Redes Neuronales, en el caso del paradigma profundo es necesario optimizar millones de conexiones por ello se incluyen otras técnicas como el *Momentum*. La idea detrás de este método es usar promedios ponderados exponencialmente de las correcciones del gradiente y luego usarlos para las actualizaciones de pesos. Matemáticamente m es actualizado mediante:

$$m_{i+1} = \alpha m_i + \eta \nabla \mathscr{N}(x, W) \tag{2.23}$$

donde ∇ representa el gradiente y el parámetro regulador $\eta = 0.9$, generalmente se define $m_0 = 0$. A partir de este método se ajustan los parámetros mediante:

$$\Delta W = W - \alpha\sigma(\mathbf{x})m - \alpha(1-\sigma(\mathbf{x}))\nabla\mathscr{N}(x, W). \tag{2.24}$$

Partiendo de este principio se proponen diversas propuestas basadas en el *Momentum*, de entre ellas destaca el *Adaptive Moment Estimation* ADAM [53], el cual a grandes rasgos combina las características del *Momentum* y RMSProp [92], utilizando gradientes al cuadrado para ajustar m, el Algoritmo describe el funcionamiento general.

Algoritmo 1 *Adaptive Moment Estimation* [53].

Require: $\alpha, \varepsilon \in \mathbb{R}$, $\beta_1, \beta_2 \in [0, 1)$

Ensure: w_i

1: **while** w_i not converged **do**
2: $\quad i \leftarrow i + 1$
3: $\quad m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot \nabla_i \mathscr{N}(x, W)$
4: $\quad v_i \leftarrow \beta_2 \cdot v_{i-1} + (1-\beta_2) \cdot \nabla_i^2 \mathscr{N}(x, W)$
5: $\quad \alpha_i \leftarrow \alpha_{i-1} \cdot \frac{\sqrt{1-\beta_2^i}}{1-\beta_1^i}$
6: $\quad w_i \leftarrow w_{i-1} - \frac{a_i \cdot m_i}{\sqrt{v_i}+\varepsilon}$
7: **end while**

donde α es la tasa de aprendizaje recomendada con un valor de 0.001, β_1 y β_2 se recomiendan con valores de 0.9 y 0.999 respectivamente y el hiperparámetro ε es recomendado en 10^{-8}. A pesar de que ADAM es el algoritmo de optimización más utilizado dentro de los *frameworks* y librerías de programación de Aprendizaje Profundo, otras variantes derivadas han sido propuestas tal como AdaMax [53], Nesterov Accelerated Adaptive Moment Estimation (NADAM) [29]. Una variante propia propuesta es el algoritmo Adaptive Noise Moment ADANM [70], el cual incluye un factor estocástico al gradiente a partir de:

$$m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot \left(\nabla \mathscr{N}(x,W) + \sigma_i^2\right) \tag{2.25}$$

$$v_i \leftarrow \beta_2 \cdot v_{i-1} + (1-\beta_2) \cdot \left(\nabla \mathscr{N}(x,W) + \sigma_i^2\right)^2 \tag{2.26}$$

$$\sigma_i^2 = \frac{\eta}{(1+i)^\gamma} \cdot \nabla \mathscr{N}(x,W) \tag{2.27}$$

donde se recomienda para los parámetros $\eta = 0.01, 0.3, 1.0$ y $\gamma = 0.55$.

2.2.1 Algoritmo general para capas neuronales profundas

En el paradigma de las Redes Neuronales Artificiales, un modelo neuronal se entiende como un conjunto de algoritmos agrupados en capas que realizan una tarea específica [30]. En la generalidad, sea una capa de extracción, ajuste o recurrente, una capa neuronal consta de tres elementos principales:

- Pesos sinápticos: sea X un conjunto de datos estructurado en tensores de un tamaño dado, existirá un peso W por cada X tal que $\forall x \in X, \exists w \in W : |X| = |W| \wedge x \neq \emptyset$.
- Producto: sea $\mathscr{N}(\cdot,\cdot)$ una función de producto de dos magnitudes euclidianas (vectores, matrices o tensores) [71], se define como una operación de producto algebraico lineal de orden n acorde a la dimensionalidad de X y W. Por ejemplo, para $X \in \mathbb{R}^{m \times n}$ la operación de producto será una convolución 2D, aunque no limitada por esta. Los parámetros de dicha función son $\mathscr{N}(X,W)$ que generan una salida h continua.
- Función de activación: $f(h)$ se entiende por una función lineal, no lineal, binaria o probabilística que escala el producto de las entradas y pesos. La salida y puede ser discreta o continua, dependiendo de la función de activación asignada y el fin de la aplicación [37].

Los anteriores componentes de la capa neuronal son utilizados por dos algoritmos esenciales para la totalidad de los tipos de capas existentes: propagación y retropropagación. De forma resumida, el algoritmo de propagación se muestra en el Algoritmo 2.

Algoritmo 2 Propagación.

Require: $e \in \mathbb{Z}, X \in \mathbb{R}^+$

Ensure: y

1: $W \leftarrow random([-1,1])$
2: **while** no converge **do**
3: **for** i$\leftarrow$ 0 **to** i=n **do**
4: $h_i \leftarrow w \times x_i$
5: $y \leftarrow \sigma(h_i)$
6: **end for**
7: $e \leftarrow e+1$
return y
8: **end while**

Como se observa en el Algoritmo 2, el algoritmo se encuentra en un ciclo de convergencia donde se incrementan las épocas e, esto ya que es implementado en dos fases: entrenamiento y predicción. Para el caso de la predicción se espera que el modelo ya haya sido entrenado, por lo que se descarta e y el ciclo de convergencia, obteniendo una respuesta y para un patrón de entrada X. En el caso del entrenamiento los algoritmos de propagación y retropropagación interactúan para ajustar las W y obtener el menor error de predicción.

Ahora se le entenderá como algoritmo de entrenamiento al algoritmo de retropropagación. Para ajustar los pesos W es necesario conocer el error empírico generado por la propagación en etapa de entrenamiento, esto basado en las salidas esperadas y' también conocidas como *ground truth*. El error puede ser calculado mediante una medida de distancia euclidiana, la cual puede ser el Error Cuadrático Medio (2.28), aunque dependiendo de la aplicación puede ser considerada una medida euclidiana absoluta, polinomial o radial a la cual se le conoce como función de pérdida. El objetivo principal del entrenamiento es reducir el error aproximándose a 0, es decir que la salida obtenida y sea igual a la esperada y'.

$$L(W) = \sum_{i=1}^{I} ||\sigma(h_i) - y_i'||^2 \tag{2.28}$$

El método general para minimizar el error es el actualizar los parámetros de manera iterativa al sumar un incremento ΔW al valor actual: $W := W + \Delta W$. Por el contrario si se utiliza una función $\mathscr{N}(x,W)$ para aproximar los valores de salida y es diferenciable con respecto a W es posible usar como algoritmo de aprendizaje el método de Gradiente Descendente:

$$\Delta W = -\alpha \frac{\partial E(W)}{\partial W} \tag{2.29}$$

donde $0 < \alpha < 1$ es un parámetro conocido como tasa de aprendizaje. En un modelo completo, las capas ocultas son actualizadas mediante la derivada de la función de activación, por ejemplo las capas activadas por la función sigmoidal (2.30) son ajustadas mediante (2.31). El algoritmo se detiene cuando el error alcanza un mínimo valor deseado.

$$\sigma(h) = \frac{1}{1+e^{(-h)}} \tag{2.30}$$

$$\frac{d\sigma(h)}{dx} = \sigma(h)(1-\sigma(h)) \tag{2.31}$$

Por lo que el algoritmo de entrenamiento es compuesto por la combinación de la propagación y retropropagación, el cual es resumido en el Algoritmo 3.

Algoritmo 3 Entrenamiento: propagación y retropropagación.

Require: $e \in \mathbb{Z}, \alpha \in \mathbb{R}, X \in \mathbb{R}^+$

Ensure: y

$W \leftarrow random([-1,1])$
while converge **do**
  for i$\leftarrow$ 0 **to** i=n **do**
    $h_i \leftarrow w \times x_i$
    $y_e \leftarrow \sigma(h_i)$
  end for
  $L(W)_e = \sum_{i=1}^{I} ||y_e - y'_e||^2$
  $m_{j+1} = \alpha m_j + \eta \nabla L(W)$
  $\Delta W = W - \alpha(y_e \cdot m_j) - \alpha(1-y_e)\nabla L(W)_e$
  $e \leftarrow e+1$
end while
  return y

Como se observa en el Algoritmo 3, existe un parámetro ajustable m el cual representa al *Momentum*, método utilizado en conjunto a la tasa de aprendizaje α para regular y acelerar el ajuste de W, optimizando el gradiente de entrenamiento y evitando mínimos locales. La idea detrás de este método es usar promedios ponderados exponencialmente de las correcciones del gradiente y luego usarlos para las actualizaciones de pesos. Matemáticamente m es actualizado mediante:

$$m_{i+1} = \alpha m_i + \eta \nabla \mathscr{N}(X,W) \tag{2.32}$$

donde ∇ representa el gradiente y el parámetro regulador $\eta = 0.9$, generalmente se define $m_0 = 0$. A partir de este método se ajustan los parámetros mediante:

$$\Delta W = W - \alpha\sigma(h)m - \alpha(1-\sigma(h))\nabla\mathscr{N}(X,W). \tag{2.33}$$

Así, el algoritmo general se resume de forma gráfica en la Figura 2.3.

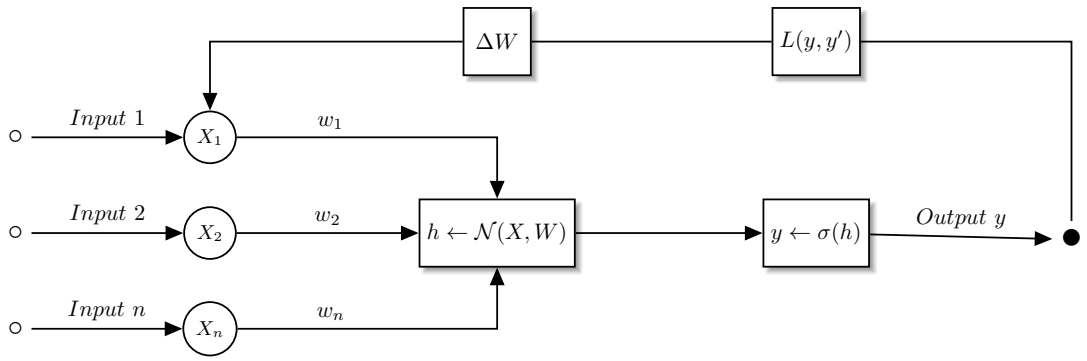


Figura 2.3 Formalización general de la capa neuronal.

2.3 Agregación Difusa

Las operaciones de agregación en conjuntos difusos son operaciones mediante las cuales se combinan varios conjuntos difusos de una manera controlada para producir un solo conjunto difuso [95]. La Agregación Difusa se puede definir como una función de transformación:

Definición 1. *Función* $f : [0,1]^\eta \to [0,1]$ *es una función de agregación si y solo si cumple las siguientes propiedades:*

1. $f(x,\ldots x) = x$ *(identidad).*
2. $f(0,\ldots,0) = 0$ *y* $f(1,\ldots,1) = 1$ *(Condiciones de borde).*
3. ***Si*** $(x_1,...,x_\eta) \leq (y_1,...,y_\eta)$ ***entonces*** $f(x_1,...,x_\eta) \leq f(y_1,...,y_\eta)$ *(No decreciente).*

La función de agregación depende principalmente del cálculo de Medidas Difusas (FM) que operan y transforman al conjunto X de entrada en un valor escalar.

Medidas difusas

A partir de la siguiente definición [80]:

Definición 2. *Una Medida Difusa μ en un conjunto X con cardinalidad η es una función de conjunto $\mu : 2^X \to [0,1]$ que cumple las siguientes propiedades:*

1. $\mu(\emptyset) = 0$, $\mu(X) = 1$, *(condición límite).*
2. $A \subseteq B$ *implica* $\mu(A) \leq \mu(B)$, *para todo* $A, B \subset X$ *(monotonicidad).*

se entiende que una FM es una función que transforma $\mu : x \to \mathbb{R}$ siendo $x \in X$. Se le llama FM regular si y solo si $\mu(X) = 1$, es decir que la totalidad del universo suma 1 al ser transformado [90]. Otra variante de FM es la Medida λ de Sugeno, la cual se define como:

Definición 3. *Dado* $X = \{x_1, \ldots, x_n\}$ *y* $\lambda \in (-1, \infty)$, *la Medida* λ *Sugeno es una función* $\mu : 2^X \to [0,1]$ *que cumple las siguientes propiedades:*

1. $\mu(X) = 1$.
2. *Si* $A, B \subseteq X | A \cap B = \emptyset$ *entonces* $\mu(A \cup B) = \mu(A) + \mu(B) + \lambda\mu(A)\mu(B)$

A la vez, λ *cumple la siguiente restricción:*

1. $\lambda + 1 = \prod_{i=1}^{\eta}(1 + \lambda\mu_i)$

Otra variación es la FM k-aditiva, la cual limita la interacción entre los subconjuntos $E \subseteq X$ a un tamaño $|E| = k$. Esto reduce el número de variables necesarias para definir la FM, y como k tomar valores en $[1, |X|]$ (para 1 la FM es aditiva), resulta un modelado simple. Independientemente de la variante, toda FM respeta las siguientes propiedades:

- Aditiva: $\forall E, F \in X | E \cap F = \emptyset : \mu(E \cup F) = \mu(E) + \mu(F)$.
- Supermodular: $\forall E, F \in X : \mu(E \cup F) + \mu(E \cap F) \geq \mu(E) + \mu(F)$.
- Submodular: $\forall E, F \in X : \mu(E \cup F) + \mu(E \cap F) \leq \mu(E) + \mu(F)$.
- Superaditiva: $\forall E, F \in X | E \cap F = \emptyset : \mu(E \cup F) \geq \mu(E) + \mu(F)$.
- Subaditiva: $\forall E, F \in X | E \cap F = \emptyset : \mu(E \cup F) \leq \mu(E) + \mu(F)$.
- Simétrica: $\forall E, F \in X : |E| = |F| \to \mu(E) = \mu(F)$.
- Booleana: $\forall E \in X : \mu(E) = 0 \vee \mu(E) = 1$.

Cuando se usa una FM para definir una función como la integral de Sugeno o la integral de Choquet, estas propiedades serán cruciales para comprender el comportamiento de la función. Por ejemplo, la integral de Choquet con respecto a una FM aditiva se reduce a la integral de Lebesgue. En casos discretos, una medida difusa simétrica dará como resultado el operador de promedio ponderado ordenado (OWA). Las medidas difusas submodulares dan como resultado funciones convexas, mientras que las medidas difusas supermodulares dan como resultado funciones cóncavas cuando se utilizan para definir una integral de Choquet.

2.3.1 Integral difusa

En la práctica una FM sirve para definir una función de agregación que siguen ciertas propiedades según se busque en la aplicación. Para combinar dos conjuntos y compuesto de FM no aditiva, se define la integral de Choquet:

Definición 4. *Sea X un conjunto de referencia con cardinalidad* η *y sea* μ *una medida difusa en X; entonces, la integral de Choquet de una función* $f : X \to \mathbb{R}^+$ *con respecto a la medida difusa* μ *se define por:*

$$Choquet(f) = \sum_{i=1}^{\eta} [f(x_{s(i)}) - f(x_{s(i-1)})] \cdot \mu(A_{s(i)}) \tag{2.34}$$

donde $f(x_{s(i)})$ *indica que los índices se han permutado de modo que* $0 \leq f(x_{s(1)}) \leq \cdots \leq f(x_{s(\eta)}) \leq 1$, $f(x_{s(0)}) = 0$ *y* $A_{s(i)} = \{x_{s(i)}, \ldots, x_{s(\eta)}\}$.

La integral de Choquet puede ser descrita de forma general en (2.35) donde $s \subseteq X$.

$$\int f \circ \mu := \int_0^{\infty} f(s|\mu(s) \geq x)\mu(x)dx \tag{2.35}$$

Otra variante de integral difusa para agregación es la integral de Sugeno, que se basa en funciones de pertenencia para conjuntos transformados al dominio difuso. La integral de Sugeno se define como:

Definición 5. *Sea X un espacio medible y* $h : X \to [0,1]$ *una función medible, la Integral de Sugeno sobre el conjunto difuso* $\tilde{A}$ *de la función h con respecto a la FM* μ*, es definida como:*

$$\int_A h(x) \circ \mu = \int_X [h_A(x) \wedge h(x)] \circ \mu \tag{2.36}$$

donde $h_A(x)$ *es la función de pertenencia del conjunto difuso* $\tilde{A}$.

Discusión

Este capítulo presentó un panorama general sobre los temas relacionados con la investigación, abarcando los conceptos básicos de estructuras tensoriales, redes neuronales de aprendizaje profundo y agregación difusa. La información presentada en este capítulo es requerida para el entendimiento de la creación de la capa de agregación neurodifusa propuesta. Para dar sustento a la teoría propuesta, en el siguiente capítulo se realiza un análisis de los trabajos relacionados a esta investigación.

Capítulo 3

Estado del Arte

En esta sección se presenta brevemente los trabajos más relevantes relacionados con la implementación y formulación teórica de Integrales Difusas, Medidas Difusas, redes Neurodifusas para fusión de datos. Esta revisión es un compilado de las aportaciones más relevantes para la investigación, así como un apartado del estado de la práctica que expone las aplicaciones patentadas relacionadas a la investigación. De la misma manera se presentan los trabajos que anteceden a este, dentro del mismo centro de investigación.

Aprendizaje de parámetros y aplicaciones de la integral de inclusión-exclusión para la fusión y análisis de datos [44]

Este artículo se concentra en la integral de inclusión-exclusión, que es una generalización de la integral de Choquet, y detalla los métodos para aprender los diversos parámetros, dada su arquitectura unificada a redes neuronales. Se valida el rendimiento y la utilidad de este enfoque en algunos conjuntos de datos de referencia. Para su implementación, la entrada x se normaliza utilizando una función sigmoide parametrizada:

$$s(x_j) = \frac{1}{1 + e^{a_j x_j + b_j)}} \tag{3.1}$$

donde a_j y b_j son parámetros que pueden optimizarse con respecto al conjunto de entrenamiento. Se puede reducir el número de parámetros restringiendo el tamaño de los subconjuntos incluidos. El concepto de k-medidas difusas aditivas permite considerar solo subconjuntos con hasta k elementos. El ajuste de parámetros es dado por:

$$\mu = \arg\min[(\sum_{A \subseteq N} \mu_A \cdot I_\lambda(x|A)) - y_i]^2 \tag{3.2}$$

donde w es el vector de variables de decisión, x es la matriz de entrada $m \times n$, y es el vector de salidas y sujeto a las restricciones $A^T w \geq b$.

PanoraMIS: un conjunto de datos de imagen de campo de visión ultra amplio para la estimación de movimiento de robot basada en visión [8]

Este artículo presenta un nuevo conjunto de datos de imágenes de campo de visión ultra amplio con una verdad precisa del terreno, llamado PanoraMIS. El conjunto de datos cubre un amplio espectro de cámaras panorámicas (catadióptrico, ojo de pez gemelo), plataformas robóticas (robots con ruedas, aéreos e industriales) y entornos de prueba (interiores y exteriores), y es muy adecuado para validar rigurosamente un nuevo robot basado en imágenes. los algoritmos de estimación de movimiento, que incluyen odometría visual, SLAM visual y métodos basados en aprendizaje profundo. En la Tabla 3.1 se muestra un resumen de bases de datos del estado del arte mencionadas en esta publicación.

Tabla 3.1 Descripción general de los conjuntos de datos de imágenes relacionados en comparación con PanoraMIS [8].

Bases de datos	Camaras	Portador	Entorno	Nube 3D	Sensores	Estadísticas
Ford Campus	Omnidirectional multicámara	Automóvil	Exteriores	Si	3-D/2-D LiDAR, IMU, GPS	2 seg, $\approx$ 10 km
AMUSE	Omnidirectional multicámara	Automóvil	Exteriores	No	IMU, GPS, sensores de altura	7 seg, 24.4 km
KITTI [56]	Estéreo	Automóvil	Exteriores	Si	IMU, GPS	22 seg, 39.2 km
IPDS	Perspectiva, ojo de pez, catadióptrica	Carro eléctrico	Exteriores	No	IMU, GPS	6 seg, 18 km
Málaga Urban [10]	Estéreo	Automóvil	Exteriores	Si	MU, GPS, LiDAR	15 seg, 36.8 km
Oxford RobotCar [63]	Ojo de pez, estéreo trinocular	Automóvil	Exteriores	Si	3-D/2-D LiDAR, IMU, GPS	1010 km
Complex Urban [50]	Estéreo	Automóvil	Exteriores	Si	3-D/2-D LiDAR, IMU, GPS, giroscopio óptico	19 seg, 191 km
New College	Multicámara omnidireccional, estéreo	Robot con ruedas	Exteriores	Si	MU, GPS, 2-D LiDAR	9 seg, 2.2 km
Rawseeds	Trinocular, perspectiva, catadióptrica	Robot con ruedas	Interiores y exteriores	No	IMU, GPS, 2-D LiDAR, odometría	$>$ 1.9 km
Marulan	Perspectiva	Robot con ruedas	Exteriores	No	2-D LiDAR, cámara IR, radar	40 seg
TUM RGB-D	RGB-D	Robot con ruedas, a mano	Interiores	Si	MoCap	39 seg
UMich NCLT [15]	Omnidirectional multicámara	Robot Segway	Interiores y exteriores	Si	MU, GPS, 3-D/2-D LiDAR	27 seg, 147.3 km
EuRoC MAV [14]	Estéreo	MAV Hexarotor	Interiores	Si	MoCap, IMU	11 seg, 0.9 km
Zurich Urban MAV [65]	Perspectiva	Quadrotor MAV	Exteriores	Si	IMU, GPS	1 seq, 2 km
LaFiDa [93]	Ojo de pez trinocular	Yelmo	Interiores y exteriores	No	MoCap, 2-D LiDAR	6 seg, 190 m
PennCOSYVIO [74]	Perspectiva, ojo de pez, estéreo	Aparejo de mano	Interiores y exteriores	No	IMU, etiquetas visuales	4 seg, 0.6 km
TUM VI [83]	Estéreo con lentes de ojo de pez	Aparejo de mano	Interiores y exteriores	No	IMU, MoCap	28 seg, 20 km
Multi-FoV [100]	Perspectiva, ojo de pez, catadióptrica	Ninguno	Interiores y exteriores	No	Datos sintéticos	2 seg

Una revisión sobre los desafíos del robot autónomo móvil y los métodos de fusión de sensores [3]

Este estudio presenta la literatura actual, los desafíos que enfrenta el robot móvil. También se presenta un estudio exhaustivo sobre dispositivos / sensores y técnicas de fusión de sensores prevalentes desarrolladas para abordar problemas como la localización, la estimación y la navegación en robots móviles en los que se organizan según su relevancia, fortalezas y debilidades. La utilidad principal de este estudio es resumir las técnicas más utilizadas para la fusión de sensores en aplicaciones prácticas, la cual es resumida en la Tabla 3.2.

Tabla 3.2 Trabajos relacionados de diferentes algoritmos de fusión de sensores [3].

Algoritmo de fusión	Clasificación del método de fusión	Sensores
Filtro de Kalman [64]	Estimación de estados	Sistema de navegación visual y sistema de navegación inercial
Filtro de partículas [76]	Estimación de estados	Cámara óptica, sonar y odometría
Redes Bayesianas	Decisión	Detección y rango de luz (LiDAR)
Dempter-shafer	Decisión	Visión y codificador

VIFB: un punto de referencia de fusión de imagen visible e infrarroja [98]

En este artículo se presenta un punto de referencia de fusión de imágenes visibles e infrarrojas (VIFB) que consta de 21 pares de imágenes, una biblioteca de códigos de 20 algoritmos de fusión y 13 métricas de evaluación. También se llevaron a cabo extensos experimentos dentro del punto de referencia para comprender el rendimiento de estos algoritmos. Al analizar los resultados cualitativos y cuantitativos, se identificaron algoritmos efectivos para una fusión de imágenes robusta y brindamos algunas observaciones sobre el estado y las perspectivas futuras de este campo.

En resumen, este artículo sirve como base para conocer las bases de datos existentes para fusión de imágenes y las métricas que se emplean en este dominio. En la Tabla 3.3 se resumen las métricas identificadas en esta publicación para evaluar cuantitativamente la fusión de imágenes, así como las principales bases de datos existentes en la literatura.

Tabla 3.3 Métricas utilizadas para medir la fusión y bases de datos de prueba.

Métricas	
Categoría	Métricas
Basado en la teoría de la información	*Cross entropy*
	entropía
	infromación mutua
	Relación pico señal-ruido PSNR [49]
Basado en similitud estructural	Medida del índice de similitud estructural SSIM
	Error Cuadrático Medio Raíz RMSE [49]
Imagen basada en funciones	Gradiente promedio [27]
	Intensidad de borde [75]
	Desviación Estándar
	Frecuencia espacial
	Rendimiento de fusión basado en gradiente
Inspirada en percepción humana	Métrica Chen-Blum [23]
	Métrica Chen-Varshney
Bases de datos	
OTB [96]	
VOT[54]	
Base de datos térmica de color de OSU	
Conjunto de datos de fusión de imagen TNO [33]	
VLIRVDIF [33]	

Algunos nuevos métodos intuitivos de fusión de información difusa en la toma de decisiones con interacción entre atributos [69]

En este artículo se proponen dos nuevos operadores integrales difusos intuitivos de Atanassov: el operador de promediación difusa de Zhenyuan de Atanassov y el operador geométrico difuso de Zhenyuan de Atanassov. Los operadores diseñados consideran la importancia de las interacciones entre los diferentes atributos y son muy adecuados para manejar problemas en los que los atributos son interdependientes o interactivos. De forma resumida, los operadores de agregación difusa intuicionista se definen como:

$$\int \beta d\eta = IFZA[\beta(y_1),...,\beta(y_n)] = \max\{\oplus_{j=1}^{2^n-1}\lambda_j\eta(E_j)|\beta(y_i) = \oplus_{j=1}^{2^n-1}\lambda_j E_j\} \quad (3.3)$$

donde $\lambda_j = (t_{\lambda_j}, f_{\lambda_j})$ es un *Atanassov intuitionistic fuzzy Zhenyuan averaging* AIFV.

Aprendizaje con funciones de activación difusas tipo 2 para mejorar el rendimiento de las redes neuronales profundas [6]

En este artículo se propone una nueva capa de activación basada en intervalos difusos de tipo 2 (IT2). Esta capa se compone de unidades de rectificación difusa (FRU) de entrada única IT2 (SIT2) para mejorar el rendimiento de aprendizaje de las redes neuronales profundas. La llamada IT2-FAL también da solución al problema del desvanecimiento del gradiente y consigue la convergencia en menor número de iteraciones de entrenamiento, ya que puede llevar la activación media a alrededor de cero. Se realizaron estudios experimentales comparativos en los conjuntos de datos de referencia MNIST, Quickdraw Pictionary y CIFAR-10. La IT2-FAL propuesta se compara con las funciones de activación Unidad lineal rectificada (ReLU), la ReLU paramétrica (PReLU) y la Unidad lineal exponencial (ELU). El resumen de la experimentación muestra una mejoría media del 18.8% con respecto a las demás funciones de activación. Adicionalmente se propone un modelo que incluye capas de convolución para extracción de información visual, este modelo se compone de tres bloques de procesamiento alternando convolución y las capa propuesta. El modelo puede ser visualizado en la Figura 3.1.

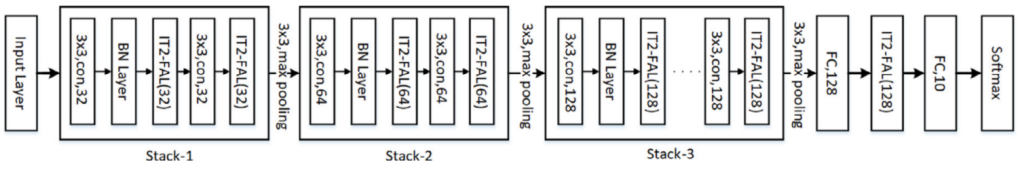


Figura 3.1 Modelo S-DNN-2 empleado para la clasificación CIFAR-10 [6].

Clasificación de conjunto conducido integral difuso utilizando medidas difusas *a priori* [1]

En este artículo se propuso el operador de agregación *Recursive Average* RAV que permite la fusión intuitiva de datos con respecto a un FM determinado. Con un operador de fusión alternativo en su lugar, se define el concepto de FM *A Priori* que se generan en base a información externa (por ejemplo, precisión de clasificación) y, por lo tanto, proporcionan una alternativa a los enfoques tradicionales de aprendizaje o especificación manual de FM. El algoritmo FM *a priori* propuesto se describe en el Algoritmo 4.

Algoritmo 4 Algoritmo FM *a priori* [1].

Require: $Acc \in \mathbb{R}^+, N_j \in \mathbb{R}$

Ensure: $g_{ap} : 2^x \to [0,1]$

$MaxAcc \leftarrow \arg\max[Acc]$
for $i \in nm(A)$ **do**
  $nm(A_i) \leftarrow 1 - \frac{MaxAcc - Acc_i}{MaxAcc - N_j}$
  if $|A| = 1$ **then**
    $g_{ap}(A_i) \leftarrow nm(A_i)$
  else
    if $nm(A_i) < nm(B_i)$ **then**
      $g_{ap}(A_i) \leftarrow nm(B_i)$
    else
      $g_{ap}(A_i) \leftarrow nm(A_i)$
    end if
  end if
end for

donde Acc son los valores de precisión de clasificación, N_j un factor de normalización y $g_{ap}(\cdot)$ una Medida Difusa.

Fusión de redes neuronales convolucionales múltiples usando Integral difusa mejorada para el reconocimiento de emociones faciales [58]

En este artículo se propusieron múltiples redes neuronales convolucionales que utilizan Integral difusa mejorada (MCNNs-IFI) para reconocer las emociones faciales. Las CNN múltiples combinadas con una integral difusa mejorada, en la que su valor de densidad difusa se optimiza mediante la optimización de enjambre de partículas (PSO), supera el inconveniente de la decisión mayoritaria en el método de votación tradicional. De forma general, el sistema recibe de entrada $X = \{x_1, x_2, ..., x_i\}$ como un conjunto de i fuentes de datos, para cada fuente se tiene una red de convolución que posteriormente la información de salida es fusionada en un bloque de integración difusa. Este bloque se inicializa con parámetros de pertenencia predefinidos y es ajustado mediante una función objetivo dada por:

$$Aptitud = \frac{TP}{TP + FP} \tag{3.4}$$

Esta aptitud sirve como parámetro para la optimización de los grados de pertenencia del bloque. El esquema general de la propuesta se muestra en la Figura 3.2, la cual se asemeja a una opción de la propuesta del propio tema.

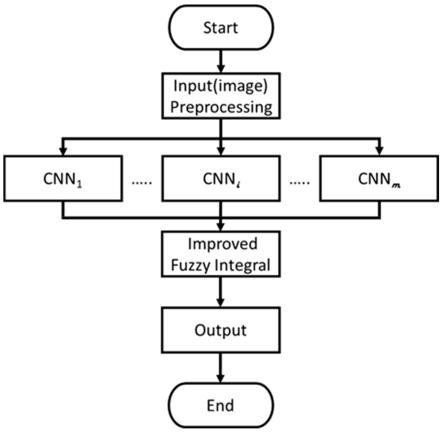


Figura 3.2 Estructura MCNNs-IFI [58].

Fusión selectiva de sensores para odometría neuronal visual-inercial [19]

En este documento se propone dos modalidades de fusión basadas en diferentes estrategias de enmascaramiento: fusión suave determinista y fusión dura estocástica, y se compara con las líneas de base de fusión directa previamente propuestas. Durante las pruebas, la red puede procesar selectivamente las características de las modalidades de sensores disponibles y producir una trayectoria a escala. Se presenta una investigación exhaustiva sobre las actuaciones en tres conducciones autónomas públicas, *Micro Aerial Vehicle* (MAV) y conjuntos de datos *Odometría Visual-Inercial* VIO portátiles.

Las bases de datos utilizadas son KITTI *Odometry dataset for autonomous driving* [38], *EuRoC dataset for micro aerial vehicle* [14], y PennCOSYVIO *dataset for hand-held devices* [74].

Permitiendo la fusión explícita en aprendizaje profundo con redes neuronales de integral difusa [48]

En este artículo se demostró que la Integral de Choquet Difusa (ChI) es una poderosa función de agregación no lineal, puede representarse como una red profunda. También se presenta un ChIMP mejorado (iChIMP) que conduce a una optimización estocástica basada en el descenso del gradiente a la luz del número exponencial de restricciones de desigualdad de ChI. Aquí se definen tres componentes principales: la Medida Difusa (FM) que es una función de transformación $g : 2^x \rightarrow \mathbb{R}^+$. Chl de una observación h sobre el universo X se define como:

$$\int h \circ g = C_g(h) = \sum_{j=1}^{N} h_{\Pi(j)}(g(A_{\Pi(j)}) - g(A_{\Pi(j-1)}))) \tag{3.5}$$

para $A_\Pi = x_{\Pi(1)},...,x_{\Pi(j)}$, $g(A_{\Pi(0)}) = 0$ y se da una permutación Π de la forma $h_{\Pi(1)} \geq h_{\Pi(2)} \geq ... \geq h_{\Pi(N)}$. El tercer componente es la Chl como $N!$ operadores de suma convexa lineal. Relativo a un ordenamiento particular de los datos i de los cuales hay $N!$ ordenamientos, la Chl se reinterpreta como:

$$f_{\Pi(i)} = \sum_{j=1}^{N} h_{\pi_i(j)}(g(A_{\pi_i(j)} - g(A_{\pi_i(j-1)})) = h_{\Pi_i}^t w_{\Pi_i} \tag{3.6}$$

3.1 Discusión del Estado del Arte

En la Sección 3 se presentaron solamente los artículos con las investigaciones de mayor relevancia e impacto sobre la investigación que se propone en este documento. En su mayoría son de utilidad para comprender los conceptos teóricos para la formulación de una capa neuronal de integración difusa, particularmente los trabajos de [99, 58, 48] muestran como utilizar una integral difusa dentro de una red neuronal, sin embargo en estos trabajos se utiliza como un procesamiento *a posteriori* a una capa de extracción o simplemente como una función de activación. En esta propuesta se pretende formular la agregación desde la FM como parámetros W, en este sentido el trabajo de [1] es de gran utilidad para interpretar las FM y adaptar el algoritmo para trabajar con características redundantes de las fuentes a fin de encontrar la relación entre ellas. Los trabajos de [58, 6] hacen referencia a bases de datos que se adaptan a este problema, por lo que resultan sumamente relevantes para la fase de experimentación. Por otro lado el trabajo de [34] muestra un método de evaluación para la calidad de fusión.

En resumen, los artículos anteriormente presentados son los más relevantes para el desarrollo de esta tesis, para un análisis más completo se recomienda la revisión del documento Anexo A (Reporte del Estado del Arte). A continuación en la Tabla 3.4 se presenta un resumen de estos destacando la utilidad de cada uno para esta experimentación.

Tabla 3.4 Discusión del Estado del Arte.

Publicación	Objetivo	Algoritmo / Técnica	Utilidad
Aprendizaje de parámetros y aplicaciones de la integral de inclusión-exclusión para la fusión y análisis de datos [44]	Proponer la combinación de la integral de inclusión-exclusión con las Redes Neuronales a fin de crear una arquitectura unificada que aprenda patrones en diversas fuentes.	Integral de inclusión-exclusión derivada de la integral de Choquet combinada con redes MLP con activación sigmoidal.	Base teórica para interpretar una integral difusa como función de activación.
PanoraMIS: un conjunto de datos de imagen de campo de visión ultra amplio para la estimación de movimiento de robot basada en visión [8]	Proponer una base de datos de imágenes de campo de visión amplio y fusionarla para control de robot móvil.	De prueba con diversos métodos y comparado con diversas bases de datos.	Muestra 19 bases de datos de múltiples sensores (ver Tabla 3.1).
Una revisión sobre los desafíos del robot autónomo móvil y los métodos de fusión de sensores [3]	Revisar diversos métodos de fusión para robots móviles.	Filtro de Kalman, Filtro de partículas, Redes Bayesianas y algoritmo *Dempter-shafer*.	Métodos de comparación (ver Tabla 3.2).
VIFB: un punto de referencia de fusión de imagen visible e infrarroja [98]	Proponer una base de datos de imágenes visibles y señales infrarrojas.	Algoritmos de fusión del Estado del Arte.	13 métricas y 5 bases de datos (ver Tabla 3.3).
Algunos nuevos métodos intuitivos de fusión de información difusa en la toma de decisiones con interacción entre atributos [69].	Formulación de métodos que buscan relaciones entre los atributos de distintos conjuntos.	Integración difusa de Atanassov.	Método de evaluación de fusión basado en matrices de decisión intuitivas.

Tabla 3.4 Discusión del Estado del Arte (continuación).

Publicación	Objetivo	Algoritmo / Técnica	Utilidad
Aprendizaje con funciones de activación difusas tipo 2 para mejorar el rendimiento de las redes neuronales profundas [6].	Introducir una nueva función de activación basada en intervalos difusos de tipo 2.	Unidades de rectificación difusa de tipo 2 en combinación con capas convolucionales.	Base para integrar funciones difusas dentro de las funciones de activación y un modelo para clasificación neurodifusa.
Clasificación de conjunto conducido integral difuso utilizando medidas difusas *a priori* [1].	Proponer un algoritmo de agregación difusa interpretable, por ejemplo usar FM proporcionada por expertos.	FM *a priori* y algoritmo RAV.	Algoritmo de inicialización de Medidas Difusas.
Fusión de redes neuronales convolucionales múltiples usando Integral difusa mejorada para el reconocimiento de emociones faciales [58].	Fusionar los mapas de características de distintas capas convolucionales mediante una integral difusa para reconocimiento visual multivista.	Arquitectura paralela de convolución fusionada mediante una integral difusa con FM ajustadas mediante optimización de enjambre de partículas.	Arquitectura de fusión y bases de datos multivista *Multi-PIE Face Database* [42] y bases con variaciones en las imágenes *CK+ Database* [59].
Fusión selectiva de sensores para odometría neuronal visual-inercial [19]	Fusión de sensores para control autónomo de vehículos aéreos y terrestres.	Fusión suave determinista y fusión dura estocástica.	Base de datos KITTI de odometría, EuRoC para vehículos micro aéreos y PennCOSYVIO para dispositivos de mano.
Permitiendo la fusión explícita en aprendizaje profundo con redes neuronales de integral difusa [48].	Similar a esta propuesta, pretende expresar una capa profunda como la integral de Choquet.	Integral difusa de Choquet.	Base teórica para formular una capa de integración difusa.

3.2 Estado de la practica

Sistemas y métodos para el control de múltiples velocidades de efectos hápticos con fusión de sensores [26]

Se describen sistemas y métodos para el control de múltiples velocidades de efectos hápticos con fusión de sensores. Un sistema ilustrativo incluye un dispositivo de salida háptica y genera una primera señal del sensor, un procesador en comunicación con el primer sensor. El procesador está configurado para: muestrear la primera señal del sensor del primer sensor a una primera velocidad, recibir una señal de referencia, determinar un error entre la primera señal del sensor y la señal de referencia, generar una señal táctil basada al menos en parte en la referencia señal y el error, transmita la señal háptica a un dispositivo de salida háptico configurado para emitir un efecto háptico basado en la señal háptica, y muestreé la primera señal del sensor a una segunda frecuencia, en la que la segunda frecuencia es menor que la primera frecuencia. El modelo operacional patentado es ilustrado en la Figura 3.3.

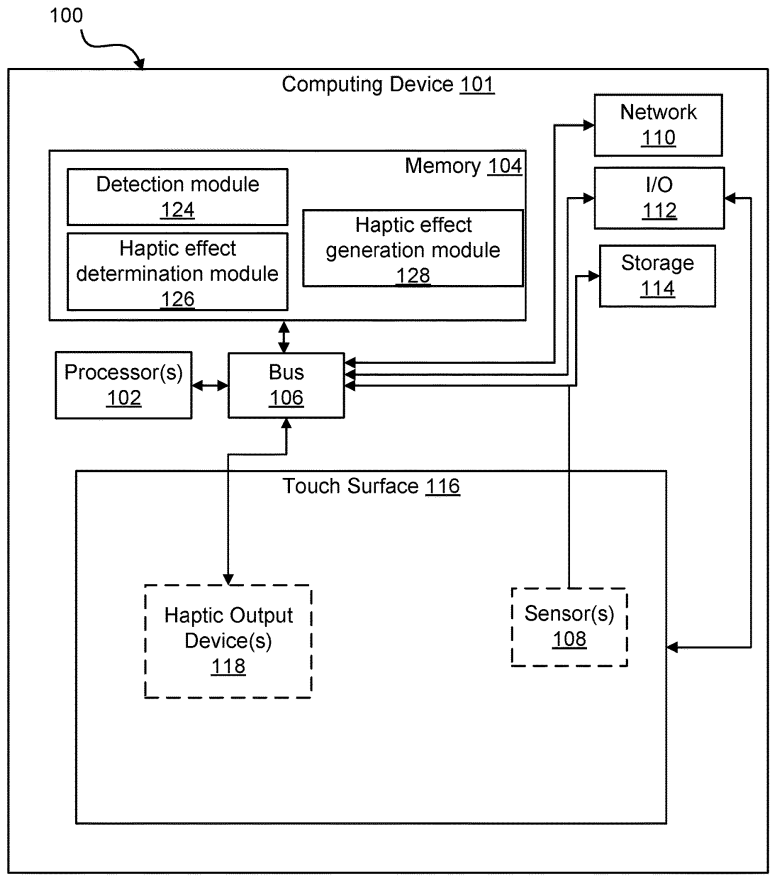


Figura 3.3 Modelo de operación patentado por [26].

La Figura 3.3 muestra un sistema ilustrativo 100 para el control de múltiples velocidades de efectos hápticos con fusión de sensores. En particular, en este ejemplo, el sistema 100 comprende un dispositivo informático 101 que tiene un procesador 102 interconectado con otro hardware a través del bus 16. El dispositivo 101 puede ser cualquier tipo de dispositivo, que incluye, por ejemplo, un teléfono móvil o almohadilla. Una memoria 104, que puede comprender cualquier medio adecuado legible por computadora tangible (y no transitorio) tal como RAM, ROM, EEPROM o similares, incorpora componentes de programa que configuran el funcionamiento del dispositivo informático. En este ejemplo, el dispositivo informático 101 incluye además uno o más dispositivos de interfaz de red 110, componentes de interfaz de entrada / salida (E/S) 112 y almacenamiento adicional 114.

Estimación de estado para vehículos aéreos que utilizan fusión multisensor [82]

Un sistema de estimación de estado que utiliza odometría visual estéreo de largo alcance que puede degradarse a un sistema monocular a gran altitud e integra mediciones de GPS, barómetro e IMU. El sistema tiene dos partes principales: un EKF que está fundido libremente y una parte de odometría visual de largo alcance. Para la odometría visual, el sistema toma la información de EKF para un seguimiento robusto de la posición de la cámara, y las salidas de odometría visual serán la medida para la actualización del estado de EKF. De forma general, en la Figura 3.4 se muestra la metodología de fusión de sensores patentada.

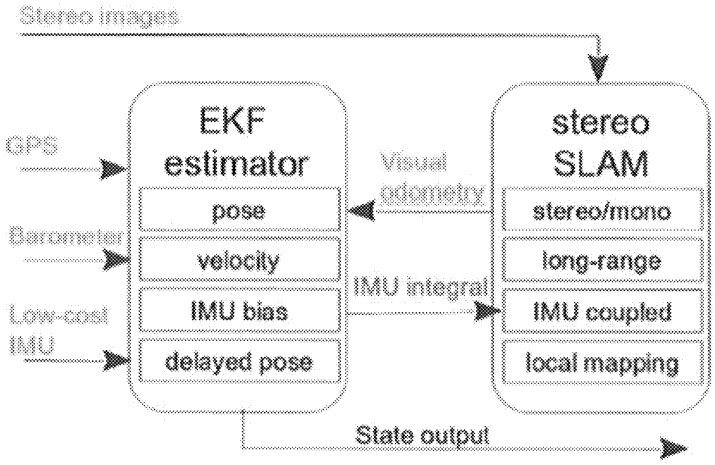


Figura 3.4 Metodología de fusión patentada por [82].

Modelo de fusión de sensores para mejorar la conciencia conversacional de la máquina [4]

Se proporcionan mecanismos, en un sistema de altavoz inteligente que tiene al menos un dispositivo de altavoz inteligente que comprende un dispositivo de captura de audio, y la lógica del sistema de altavoz inteligente, para procesar datos de muestra de audio capturados por el dispositivo de captura de audio. El dispositivo de captura de audio captura una muestra de audio de un entorno monitoreado y uno o más dispositivos sensores capturan datos del sensor que representan indicadores de atención no verbales asociados con un hablante de una parte de la muestra de audio. La lógica del sistema de altavoz inteligente evalúa los indicadores de atención no verbal de los datos del sensor para determinar si la parte del habla de la muestra de audio se dirige o no al dispositivo de altavoz inteligente. En respuesta a la determinación de que la porción de voz de la muestra de audio se dirige al dispositivo de altavoz inteligente, un sistema cognitivo asociado con el sistema de altavoz inteligente genera una respuesta a la porción de voz. La Figura 3.5 es un diagrama de ejemplo de un servicio de sensor de fusión basado en una red neuronal de acuerdo con una realización ilustrativa.

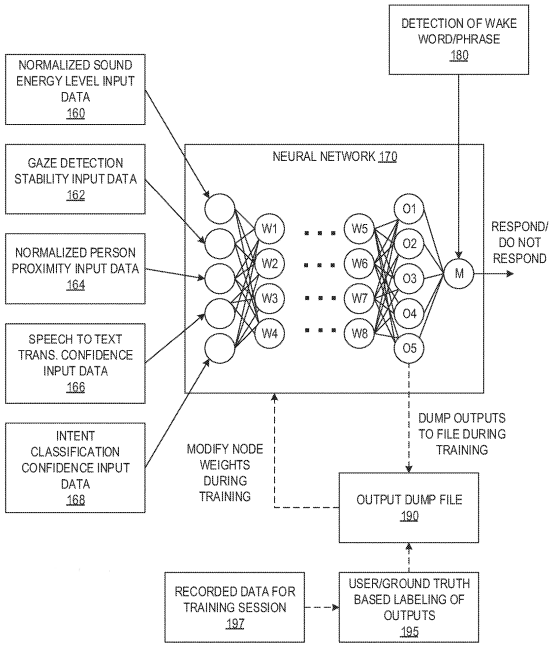


Figura 3.5 Modelo neuronal patentado [4].

Como se muestra en la Figura 3.5, varias características relevantes de los indicadores de atención no verbales 160 - 168 se introducen en la capa de entrada de la red neuronal 170, que procesa estas entradas aplicando, en los diversos nodos internos de las capas de la red neuronal 170, algoritmos para generar salidas que indican si el nodo ha evaluado

las entradas para indicar si un criterio particular ha sido satisfecho por las entradas. Las salidas de los diversos nodos se pueden ponderar de acuerdo con los pesos asignados $(W_1,\ldots,W_X)$ que se aprenden a través del entrenamiento de la red neuronal 170. En algunas realizaciones ilustrativas, esta funcionalidad puede combinarse con la detección de palabras / frases de activación 180,

Fusión de sensores habilitados por radar [39]

Este documento de patente describe aparatos y técnicas para la fusión de sensores habilitados por radar. En algunos aspectos, se proporciona un campo de radar y se reciben señales de reflexión que corresponden a un objetivo en el campo de radar. Las señales de reflexión se transforman para proporcionar datos de radar, de los cuales se extrae una característica que indica una característica física del objetivo. Según las características del radar, se activa un sensor para proporcionar datos de sensor complementarios asociados con la característica física. La función de radar se aumenta luego con los datos del sensor suplementario para mejorar la función de radar, por ejemplo, aumentando la precisión o resolución de la función de radar. Al hacerlo, se puede mejorar el rendimiento de las aplicaciones basadas en sensores, que se basan en las características mejoradas del radar. Esta aplicación es usada en sistemas de sensores de baja potencia, por ejemplo televisiones inteligentes como se muestra en la Figura 3.6, donde es necesario escanear el área para detectar personas antes de hacer reconocimiento de gestos.

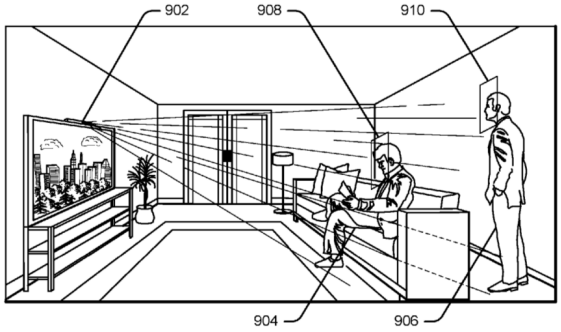


Figura 3.6 Ejemplo de fusión de sensores de baja potencia implementada para televisión que incluye un motor de fusión de sensores

Fusión de sensor de nueve ejes cuaterniones con filtro de Kalman Modificado [86]

Los cuaterniones son una extensión de los números reales, similar a la de los números complejos. Mientras que los números complejos son una extensión de los reales por la adición de la unidad imaginaria i, tal que $i^2 = -1$, los cuaterniones son una extensión generada de manera análoga añadiendo las unidades imaginarias i, j y k a los números

reales tal que:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.7}$$

los elementos 1, i, j y k son los componentes de la base de los cuaterniones considerado como un $\mathbb{R}-$espacio vectorial de dimensión 4. Dado lo anterior, en esta patente se muestra un sistema para determinar y corregir una orientación calculada de un dispositivo basado en datos de un acelerómetro y un giroscopio, como se muestra en la Figura 3.7. El sistema utiliza un filtro de Kalman modificado que actualiza la covarianza para reducir la descomposición con el tiempo en función de un residuo del filtro. El sesgo del giroscopio se rastrea y se compensa en función de la covarianza actualizada y la acción determinada desde un ángulo entre una aceleración medida vector de eración y un vector de aceleración esperado, residual giratorio. El residual se basa en una orientación observacional, un marco predicho basado en el ángulo.

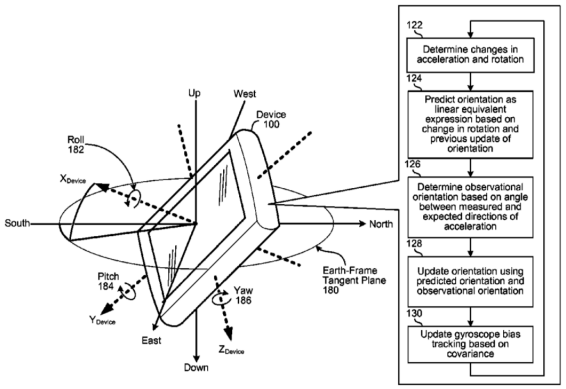


Figura 3.7 Visualización del dispositivo patentado [86].

Escaneo optimizado de objetos usando fusión de sensores [81]

En este patente se detalla como un dispositivo de pantalla montada en la cabeza (HMD) que tiene un paquete de sensores equipado con diferentes sensores para que la información es complementaria a las imágenes 2D capturadas de objetos o escenas en un entorno del mundo real se pueda utilizar para determinar una transformación optimizada de pares estéreo de imágenes y descartar datos erróneos que de otro modo evitarían exploraciones exitosas utilizadas para la construcción de un modelo 3D en, por ejemplo, aplicaciones de mundo virtual. Dicha información complementaria puede incluir una o más ubicaciones mundiales, rotación mundial, datos de imágenes de un campo de visión extendido (FOV) o datos de mapas de profundidad.

3.3 Antecedentes

Segmentación no Paramétrica de Tejidos Cerebrales Mediante una Arquitectura Paralela de Redes Neuronales Convolucionales [68]

En este trabajo se desarrolló e implementó un método de segmentación no paramétrica de tejidos cerebrales, basada en una arquitectura paralela de Redes Neuronales Convolucionales. El modelo U-Net fué utilizado para la adaptación de una arquitectura paralela. La arquitectura implementada se compone de cuatro CNNs, a las cuales les fue realizado un ajuste paramétrico, para su entrenamiento individual. Cada uno de los modelos que componen a la arquitectura paralela, lograron realizar la segmentación binaria de un tejido cerebral; la segmentación completa de la imagen evaluada se obtuvo mediante la unificación de los tejidos. Los repositorios de imágenes utilizadas fueron BrainWeb para entrenamiento y BraTS 2017 para validación y pruebas. A los estudios de resonancia magnética utilizados para entrenamiento se les aplicó un pre-procesamiento para la supresión de tejidos no blandos, seguido de una transformación a formato TIF para su uso en los framework Keras y TensorFlow; finalmente, se realizó un aumento de información para el entrenamiento de los modelos de CNN utilizados.

Sistema De Navegación Inercial Asistido Por Visión Para Robots Móviles Terrestres [72]

En este trabajo se hace fusión de señales de medición inercial IMU con información visual a fin de reducir la incertidumbre en la estimación de la posición de un robot móvil obtenida por un sistema de navegación inercial que brinda datos de posición, velocidad y dirección angular de corto plazo. Como resultado de este trabajo se deriva una heurística capaz de controlar las integraciones de los componentes x y y del INS, mediante la detección del estado ("movimiento" o "reposo") en ambos ejes por medio de datos visuales. Los experimentos se realizaron en entornos con condiciones controladas, registrando un error menor a 0.08m en trayectorias de longitud total de 1.1 m.

Sistema Embebido para Asistencia de Conducción Basado en Lógica Difusa Tipo-2 [17]

En este trabajo de tesis se present´o el dise~no y desarrollo de un asistente de conducci´on basado en L´ogica Difusa Tipo 2. El sistema propuesto adapta m´etricas a tres experimentos que evaluaron la detecci´on de un veh´ıculo frontal, detecci´on de nivel de riesgo por intenci´on en el camino y detecci´on de nivel de riesgo de veh´ıculo frontal en conjunto de la intenci´on. Para la experimentaci´on de lo anteriormente propuesto se

realiz´o con ayuda de datos Yano DD, UTA Real-life Drowsiness y TME Motorway. Los resultados que se obtuvieron fueron de estructura cualitativa y cuantitativa, mostrando que el enfoque que se hab´ıa propuesto es eficiente para la detecci´on de riesgo, pues cuenta con un rendimiento mayor a 0.90 en las m´etricas de evaluaci´on. Este trabajo se tomo como referencia para las m´etricas de evaluaci´on de la detecci´on de objetos

Conducción Autónoma de un Vehículo Simulado mediante un Modelo de Red Neuronal Convolucional Recurrente [60]

En este trabajo se propone un modelo de Red Neuronal Híbrido compuesto por capas de Convolución y Recurrentes para realizar el control de un vehículo simulado en el Sistema Operativo Robótico. El vehículo en cuestión es datado de sensores visuales y espaciales que permiten la percepción del entorno, esta información es procesada por el modelo neuronal para extraer información visual del camino y transformarla en ángulos de giro del volante. La fusión de estos sensores es realizada mediante alineación de los sensores a partir de transformaciones $(x,y,z) \mapsto (u,v)$ del mundo real a un plano de la imagen. Se utiliza la fusión como un bloque de preprocesamiento por separado del modelo neuronal.

Navegación de un robot omnidireccional basada en Lógica Difusa Tipo-2 [78]

En este trabajo se presenta el diseño, desarrollo y prueba de un sistema de navegación para un robot omnidireccional basado en Lógica Difusa Tipo-2. Este trabajo se divide en tres apartados principales: El primer apartado es la adquisición de información a través de un mapa de profundidad, utilizando una cámara estéreo, el segundo apartado es un sistema de inferencia mediante la Lógica Difusa Tipo-2 y el tercero es su aplicación sobre un robot omnidireccional con ruedas tipo *mecanum*. Los resultados cuantitativos, muestran que el control en una trayectoria generada y seguida, para llegar a una meta resulta eficiente al implementar la visión como tipo de navegación, teniendo un rendimiento mayor a 0.80 en escenarios con obstáculos estáticos y dinámicos

Detección de obstáculos y planos durante el desplazamiento vehicular [18]

En este trabajo se realiza un análisis integral para la percepción del ambiente de un vehículo autónomo, para realizar la planificación de maniobrar de forma eficiente y segura. A través de sensores como LiDAR, realiza la visión del escenario y se remueve el ruido en las nubes de puntos causado por polvo o lluvia. En combinación con un sensor RGBD, se obtiene una nube de puntos optimizada con la cual se segmenta el camino e identifica los obstáculos mediante los algoritmos RANSAC y DBSCAN. Se realizó la experimentación en el entorno de simulación de Matlab, obteniendo métricas de precisión en detección entre 0.85 y 0.95.

Capítulo 4

Metodología de Solución

4.1 Propuesta de solución

A partir de lo analizado en el Marco Teórico, en esta propuesta de tesis doctoral se pretende formular, justificar y evaluar un teorema matemático de Agregación Neurodifusa a partir de las siguientes observaciones:

- Dada la importancia de los parámetros W dentro de las Redes Neuronales dentro de funciones $\mathscr{N}(x,W)$ y que a la vez es una función de transformación $\mathscr{N}(x,W) \rightarrow \mathbb{R}$ donde W es constante para el conjunto X, es posible redefinir la Definición 2 como $\mu(\mathbf{X},W_\mu) \rightarrow [0,1], \mathbf{X}^{2\times X}$ de forma que la función μ transforme los datos de dos fuentes de forma inteligente como lo plantea [41].

- Tomando como ejemplo la integral de Choquet (2.34) y asumiendo que dado $f : X \rightarrow \mathbb{R}^+ \Rightarrow x \rightarrow \mathbb{R} | x \in X \wedge |x| = 1$, se puede redefinir como:

$$f = \sum_{i=1}^{\eta} [\mathscr{N}(x_{s(i)}, W_{s(i)}) - \mathscr{N}(x_{s(i-1)}, W_{s(i-1)})] \cdot \mu(\mathbf{X}, W_\mu) \tag{4.1}$$

  donde $x \in \mathbf{X}$ es un par vectorial x^2 y $|W_s| = \eta$.

Además se pretende que la FM permita la agregación de múltiples fuentes de datos de tal forma que $\mathbf{X}^{n\times X}$, siendo que la cardinalidad de cada fuente X sea simétrica o no. Todo lo anterior implica la formulación de un algoritmo que genere ΔW, ΔW_μ y ΔW_s a la vez a pártir de una función objetivo definida. Lo anterior se resume de forma gráfica en la Figura 4.1.

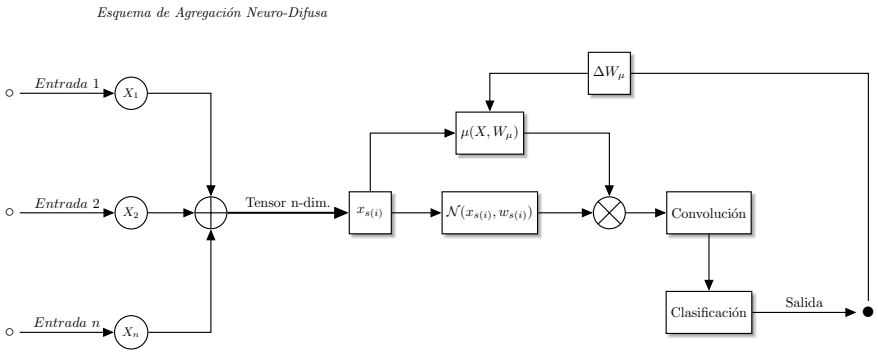


Figura 4.1 Representación gráfica de la metodología propuesta basada en la expresión (4.1).

4.2 Formulación de capas neurodifusas

A modo de resumen de las adaptaciones algorítmicas para formular la capa neurodifusa, a continuación se enumeran los componentes de la propuesta:

1. Entradas X: ya sea un tensor de orden 2 o superior [46], se permite una entrada de diferente orden para cada elemento de la primera dimensión, es decir, la señal $x_{1,}$ es estructuralmente diferente a la $x_{2,}$, permitiendo operar señales 2D y 1D en el mismo tensor.

2. Las entradas X están estructuradas en un tensor asimétrico x_s de máxima dimensionalidad $x_s \in \mathbb{R}^{d\times(m\times n)}$, según el tensor de mayor orden.

3. Pesos difusos W_μ: a partir de una medida difusa μ [80] definida como una función de transformación dentro del dominio difuso $\mu(x) : [0,1]^\eta \mapsto [0,1]$, e interpretada como una función de ponderación tal que $\mu(\emptyset) = 0$, $\mu(X) = 1$, los pesos difusos W_μ se definen para cada $x \in X$ tal que $\sum_{i=0}^{|X|} w_{\mu_i} = 1$. A su vez, existen pesos no difusos W_s para cada entrada ordenada x_s y que son ajustados por ellos, independientemente de la agregación difusa.

4. Estado oculto h: depende del operador implementado, para la generalidad de la neurona se puede utilizar una función algebraica del producto de magnitudes euclidianas [71]. Por lo tanto, la integral difusa de Choquet [9] definida como:

$$\int f \circ \mu = \sum_{i=1}^{\eta} [f(x_{s(i)}) - f(x_{s(i-1)})] \cdot \mu(x_{s(i)}) \tag{4.2}$$

donde $f(\cdot)$ indica la transformación difusa de los datos ordenados y $\mu(\cdot)$ la agregación usando las medidas difusas μ. Dado que $x \in X$ es un tensor de orden

mayor que 0, es necesaria una reducción utilizando un operador como el producto interior $\mathscr{N}(\cdot,\cdot)$, esto en relación con la introducción de pesos no difusos W_s para cada x_s. Así el estado h se detalla como la extensión de la integral de Choquet:

$$h = \sum_{i=1}^{\eta} [\mathscr{N}(x_{s(i)}, W_{s(i)}) - \mathscr{N}(x_{s(i-1)}, W_{s(i-1)})] \cdot \mu(X, W_\mu) \tag{4.3}$$

de tal forma que la integral difusa $\mu(\cdot,\cdot)$ depende del producto lineal $\mathscr{N}(\cdot,\cdot)$, que en el caso debe ser un producto exterior *veces* debido a la asimetría de los tensores de entrada.

5. Activación $\sigma(h)$: Como esta neurona no está dedicada a la clasificación, no se usa una función probabilística como Softmax [91], por lo que se debe usar una función no lineal o rectificadora como cualquier capa oculta. Algoritmo de entrenamiento: la propagación del error dentro de un modelo completo se realiza de forma normal excepto para esta capa, para la cual es necesario actualizar tanto los pesos W_s como los pesos difusos W_μ. En cuanto a Momentum, su versión difusa reside en el escalado del gradiente ∇ [85], este se modifica por el valor escalar de la integral difusa obtenida del producto de agregación:

$$m_{\mu_{i+1}} = \alpha m_{\mu_i} + \eta \nabla \mu(x_s, w_s) \tag{4.4}$$

donde $\nabla \in [0,1]$ y el Momentum difuso $m_\mu \in [-1,1]$ están restringidos. De esta forma el ajuste ΔW_μ viene dado por:

$$\Delta W_\mu = W_\mu - \alpha(y \cdot m_\mu) - \alpha(1-y))\nabla\mu(X, W_\mu). \tag{4.5}$$

Por otro lado, la función de pérdida debe permanecer en el dominio difuso, por lo que el MSE puede generar alteraciones al ajuste al poder obtener valores mayores a 1. Por ello se utiliza el Logaritmo del Coseno Hiperbólico:

$$L(W) = \sum_{i=0}^{n} log\left(\frac{e^{(y'-y)} + e^{-(y'-y)}}{2}\right) \tag{4.6}$$

En resumen, en el Algoritmo 5 se presenta el algoritmo de propagación y backpropagation de esta propuesta.

Algoritmo 5 Entrenamiento neurodifuso ΔW_μ .

Require: $e \in \mathbb{Z}, \alpha \in \mathbb{R}, X_1, X_2, X_n \in \mathbb{R}^+$

Ensure: y

1: $W_s \leftarrow random([-1,1])$
2: $W_\mu \leftarrow 0$
3: $x_s \leftarrow X_1 \times X_2 \times X_n : m \times N = \{n \in \mathbb{N} | \ |N| = m\}$
4: **while** converge **do**
5: **for** $i \leftarrow 0$ **to** $i = n$ **do**
6: $h_{s_i} \leftarrow \mathcal{N}(w_s, x_{s_i})$
7: $h_{\mu_i} \leftarrow \mu(x_{s_i}, w_\mu)$
8: $h_i = \sum_{i=1}^{\eta} [h_{s_i} - h_{s_i - 1}] \cdot h_\mu$
9: $y_e \leftarrow \sigma(h_i)$
10: **end for**
11: $L(W)_e = \sum_{i=0}^{n} \log\left(\frac{e^{(y' - \sigma(h_\mu))} + e^{-(y' - \sigma(h_\mu))}}{2}\right)$
12: $m_{\mu_{i+1}} = \alpha m_{\mu_i} + \eta \nabla \mu(x_s, w_s)$
13: $\Delta W_\mu = W_\mu - \alpha(\sigma(h) \cdot m_\mu) - \alpha(1 - \sigma(h)))\nabla \mu(X, W_\mu)$
14: $\Delta W_s = W_s - \alpha(y_e \cdot m_j) - \alpha(1 - y_e)\nabla L(W)_e$
15: $e \leftarrow e + 1$
16: **end while**
return y

En el Algoritmo 5 del paso 3, se usa la estructuración de tensor asimétrico. Esta transformación genera una estructura única que respeta cada forma de fuente de datos, agrupándose en una dimensión adicional de m. El algoritmo requiere dos cálculos de producto, uno lineal h_s y otro basado en medida difusa h_μ, posteriormente se unifican mediante la integral difusa en el paso 8 en la estructura de salida h. Cabe señalar que a partir de este algoritmo no se genera una salida y ya que esta pertenece a la parte de clasificación utilizada en el modelo neuronal a implementar, sin embargo es necesaria para el ajuste de los pesos W_s. Por lo tanto, el modelo neurodifuso se define en la Figura 4.2.

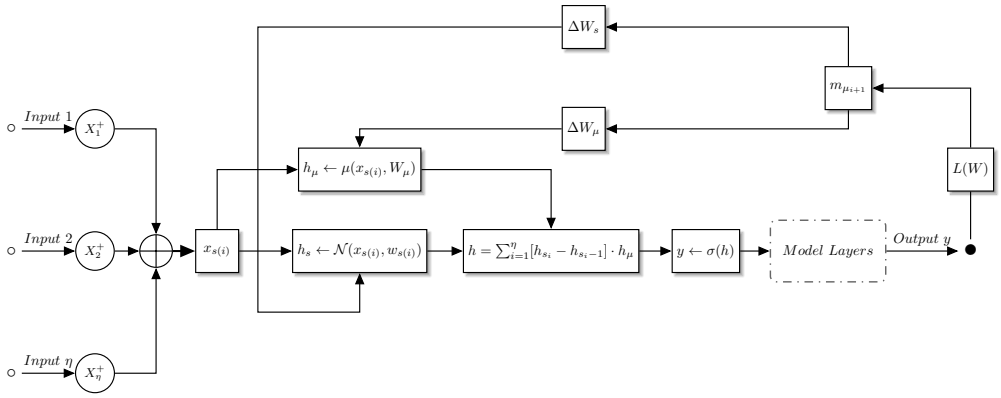


Figura 4.2 Formalización de la capa de agregación neurodifusa.

En la descripción anterior, la función de estado oculto h es capaz de realizar la agregación de dos señales, usando solo una función de activación σ rectificador. Sin embargo, para una aplicación como la conducción autónoma, el modelo requiere una mayor complejidad, así como un mayor número de capas con diferentes activaciones. Por esta razón, la Figura 4.2 representa las capas del modelo en un bloque punteado, lo que implica que después de la capa de agregación existen capas densas, convoluciones o cualquier otra que realiza diferentes tareas con información de fuentes unificadas. A continuación se detallan los recursos utilizados para adaptar el modelo de conducción autónoma con agregación neurofuzzy.

4.3 Modelo de autoconducción

El propósito de la capa de agregación difusa propuesta es combinar múltiples señales para reducir los datos dentro de un modelo neuronal. Por lo que es necesario tener un modelo de conducción para garantizar el rendimiento y posteriormente reducirlo. Para esto, se utiliza el modelo Chofer distribuido en el tiempo que se muestra en la Figura 4.3.

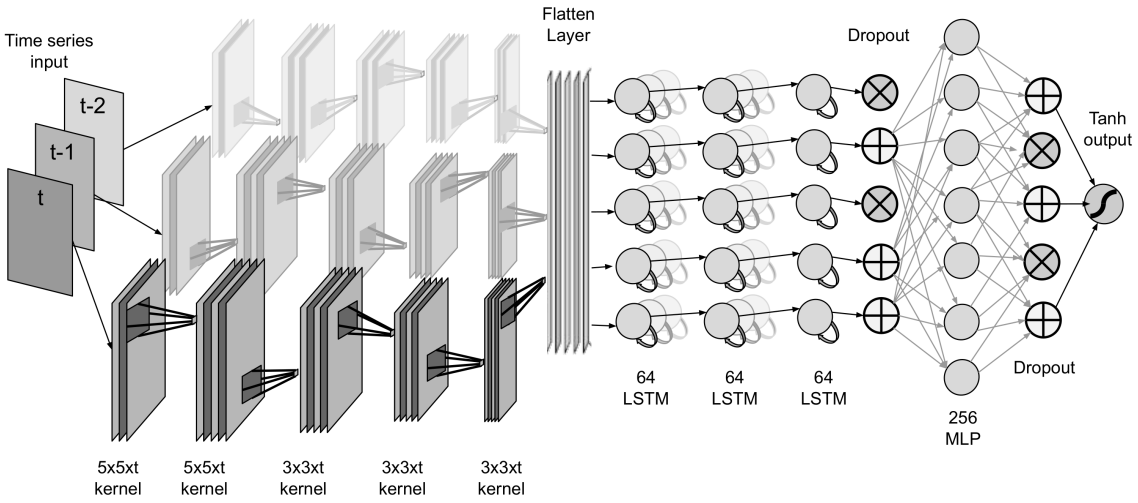


Figura 4.3 Modelo Chauffeur distribuido en el tiempo [61].

Este modelo recibe información en forma de secuencia de imágenes, en el espacio de color YUV, por lo que se estructura como un tensor 4D de tamaño $t \times 3 \times 200 \times 320$, donde t representa los fotogramas por instancia, 3 son los canales de color y 200×320 es el tamaño de cada imagen. La salida del modelo se reduce a un escalar obtenido de la función de activación de la capa de salida σ, dada por una función Tangente Hiperbólica que genera salidas en el intervalo $[-1, 1]$, estos valores representan grados de giro en la dirección del vehículo Sin embargo, el modelo td-Chauffeur no puede detectar objetos. Para llevar a cabo la detección de objetos a partir de información visual, se utilizó como base el modelo Mobilenet, que se muestra en la Figura 4.4.

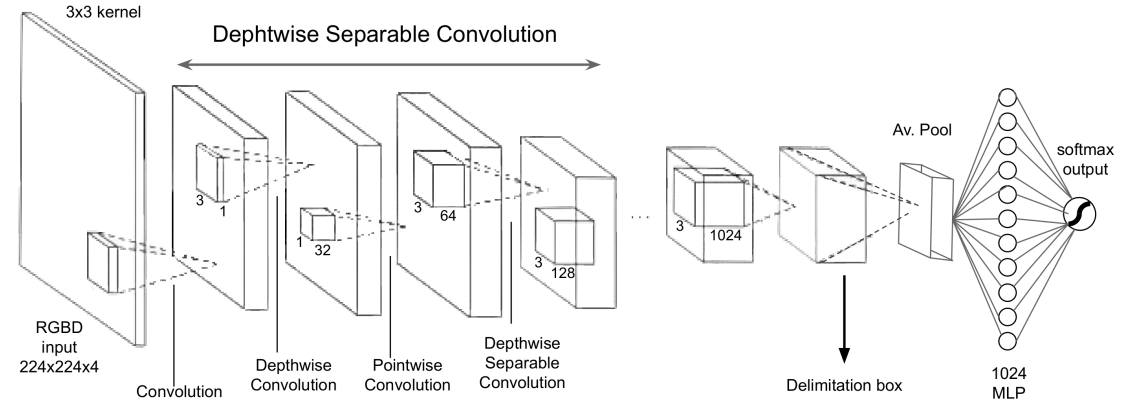


Figura 4.4 Modelo Mobilenet para la detección de objetos [45].

Este modelo móvil utiliza circunvoluciones separables en profundidad. Reduce significativamente la cantidad de parámetros en comparación con las redes con circunvoluciones regulares y la misma profundidad, lo que da como resultado una red

neuronal profunda liviana. La base del modelo es utilizar capas de convolución de profundidad y convolución de puntos, de manera que el modelo pueda adaptarse a la información a procesar. Originalmente recibe un tensor 3D de $224 \times 224 \times 3$ que representa una imagen RGB, sin embargo para esta aplicación se ha adaptado para procesar un tensor 3D de $224 \times 224 \times 4$ que se interpreta como una imagen RGBD. Como salida obtiene los cuadros delimitadores y la distancia desde el centro de los objetos.

En una metodología simple, es posible realizar la tarea ejecutando ambos modelos en diferentes hilos y procesando ambas salidas en un tercer algoritmo. La limitación de eso es el costo computacional, en una computadora con capacidad de procesamiento es posible ejecutar utilizando la mayoría de los recursos, sin embargo en un sistema embebido esto está limitado por: 1) la capacidad de procesamiento de GPU y CPU es menor que la de un computadora de escritorio, 2) el aumento del consumo de energía y calefacción debido al uso excesivo de recursos hacen que la implementación sea inviable, y 3) la sincronización de modelos es un punto a considerar incluso en una estación de trabajo. Por ello, se propone la integración de ambos modelos a través de una distribución tensorial asimétrica, generando un único modelo multifunción con capas de agregación neurodifusa para reducir la complejidad del modelo. Esta propuesta se puede ver visualmente en la Figura 4.5.

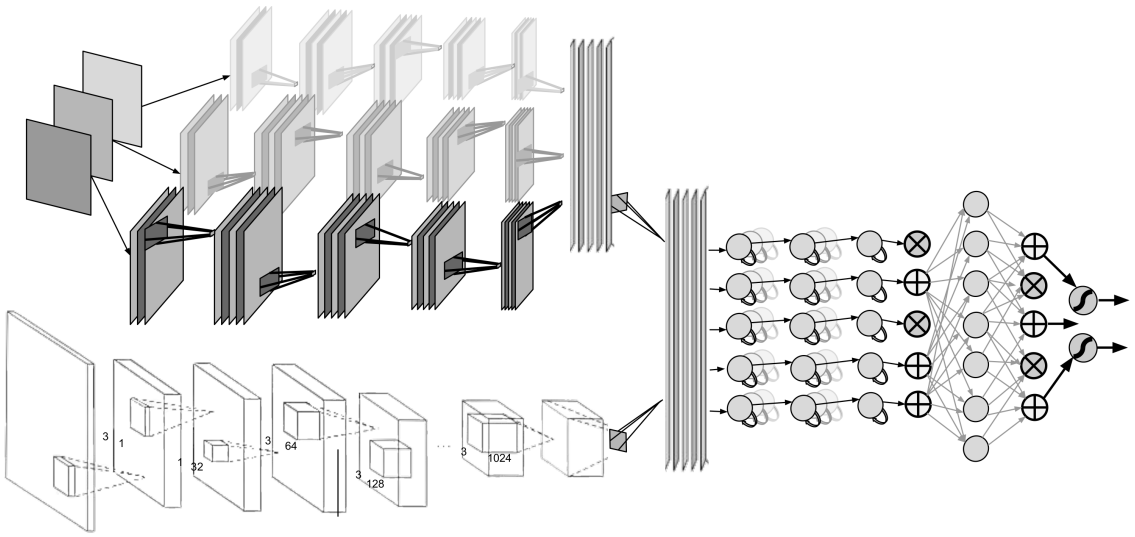


Figura 4.5 Modelo operacional de autoconducción propuesto [62].

La representación gráfica hace parecer que el modelo es una estructura paralela de dos redes neuronales. A nivel de tensor se interpreta como dos capas de la dimensión más externa n, sin embargo en la dimensión $n-1$ la forma del tensor cambia de tal

manera que la entrada se define como $I = \{\{224\ por 224 \times 3 \times t\}, \{224 \times 224 \times 4\}\}$. La asimetría de la entrada de cada capa neuronal es notoria, sin embargo la reducción de hilos de procesamiento se lleva a cabo distribuyendo el proceso GPU únicamente. Donde se observa el encogimiento del modelo es en la capa A_{W_μ} donde se obtienen los cuadros delimitadores, el encogimiento temporal de la capa *Flatten* y se suma junto con la traslación y rotación $I_p = \{\{x^t, y^t, z^t\}, \{x^r, y^r, z^r\}\}$. Finalmente, la salida se extiende a dos funciones TanH para controlar la dirección y la aceleración.

Capítulo 5

Experimentación

5.1 Diseño de experimentos

Para validar el desempeño de la propuesta se realizaron dos experimentos: una simulación en ROS con variables de un entorno urbano real y un prototipo a escala probado en un entorno controlado. En ambos casos se contemplan escenarios con obstáculos y caminos libres, así como el uso de más de un sensor. Para cuantificar el rendimiento se utilizaron métricas supervisadas y no supervisadas, especializadas en medir la calidad, precisión y autonomía de la conducción autónoma. Ambos experimentos fueron evaluados por las siguientes métricas.

5.1.1 Métricas

En primer lugar, se realiza la evaluación supervisada para conocer la similitud del modelo de conducción con las acciones realizadas por un conductor humano. Para esto, se creó una verdad de campo y' a partir de la captura de comandos de dirección durante la conducción manual en la ruta de prueba. A partir de esta referencia es posible medir la diferencia en términos de distancia entre lo que se obtiene y lo que se espera, por lo que se utiliza como métrica de evaluación el Error Cuadrático Medio, de igual forma se utiliza el Coseno Distancia:

$$cos(\theta) = \frac{\sum_{i=0}^{n} y_i \cdot y'_i}{\sqrt{\sum_{i=0}^{n} y_i^2} \cdot \sqrt{\sum_{i=0}^{n} y_i'^2}} \tag{5.1}$$

el rango de salida abarca $[-1, 1]$, donde 0 indica un error cero, por lo tanto, 1 y -1 indican un error completo a la izquierda y a la derecha. El MSE proporciona una tasa de error entre ambos tipos de conducción, independientemente de las variaciones producidas por el tiempo y la velocidad de desplazamiento del vehículo, por ello se utiliza la métrica *Driving Behavior*:

$$DB = \left\| \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})}{n}} - \sqrt{\frac{\sum_{i=1}^{n}(y'_i - \bar{y}')}{n}} \right\| \tag{5.2}$$

esta métrica evalúa todo el recorrido calculando el desvío entre ambas líneas, por lo que el tiempo no interviene en la evaluación y obteniendo así un índice de distancia invariante a la velocidad.

De manera no supervisada, la conducción se mide por Path Smoothness, esto se refiere a la amplitud de los ángulos que se describen mientras el vehículo está en movimiento:

$$\kappa = \frac{1}{n}\sum_{i=2}^{n}\left[\arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) - \arctan\left(\frac{y_{i-1} - y_i}{x_{i-1} - x_i}\right)\right] \tag{5.3}$$

donde x_i y y_i representan la posición del vehículo en un segmento de trayectoria específico. La métrica obtiene el ángulo entre dos segmentos consecutivos del camino. Un valor de suavidad más bajo indica una ruta más suave. Por otro lado, para evaluar la capacidad de operar independientemente de un conductor, se hace uso de la métrica de autonomía propuesta por [11]:

$$autonomy = \left(1 - \frac{interventions \cdot 6}{elapsed\ time}\right) \cdot 100 \tag{5.4}$$

así como la métrica Tiempo de Autonomía Absoluta [70], que miden las intervenciones de un conductor humano como un error en el sistema, siendo el tiempo de autonomía la métrica orientada a la intervención absoluta:

$$AAT = \left(1 - \frac{tiempo\ de\ intervención}{tiempo\ transcurrido}\right). \tag{5.5}$$

Para complementar la experimentación, se evaluaron en las mismas condiciones algunos métodos conocidos en la literatura para la conducción autónoma. Específicamente, fueron el modelo Pilotnet [11], la biblioteca de autoconducción Donkeycar [89], Matlab Automated Driving Toolbox [66]. Para estos se tomaron modelos de conducción autónoma y se replicaron en el sistema ROS utilizándolo como intermediario. Para complementar la experimentación, se retoma un método clásico empleado para realizar la autoconducción reactiva. Este método consta de la extracción de los bordes en la imagen del camino [84], posteriormente buscar aquellas líneas que intersecten con los ángulos de referencia que existen para las líneas del carril de conducción, esto realizado mediante la transformada Hough [20].

5.1.2 Experimento 1: Simulación de autoconducción de ROS

Esta tarea se realizó en base a los diseños de ROS de libre acceso, *CAT Vehicle Testbed*, consta de un vehículo Ford Escape con las medidas físicas reales. El paquete incluye sensores integrados en el vehículo; sin embargo, se modificaron para adaptarse a esta aplicación. Originalmente con un sensor Velodyne LiDAR con una nube de puntos de 2000, se modificó para producir una nube de puntos de 12000 con una apertura horizontal de $90°$ y una apertura vertical de $33,67°$. De la misma forma, el paquete cuenta con dos cámaras posicionadas en el vehículo en ángulos de $-45°$ y $45°$ con origen al frente del vehículo, estas capturan imágenes RGB de tamaño 800×800 pixeles . Para esta aplicación, se requería una sola cámara apuntando al origen con un tamaño de 640×320 píxeles. Para evaluar la autonomía del vehículo fue necesario integrar entornos urbanos con diferentes características como intersecciones, casas y edificios, caminos cerrados y obstáculos en el camino. Para tener una variedad de escenarios posibles, se integró la simulación de vehículos y ciudades, que representa completamente una pequeña ciudad. Se tomaron algunos fragmentos de la vía con obstáculos para realizar la evaluación de conducción autónoma. Estos se muestran en la Figura 5.1.

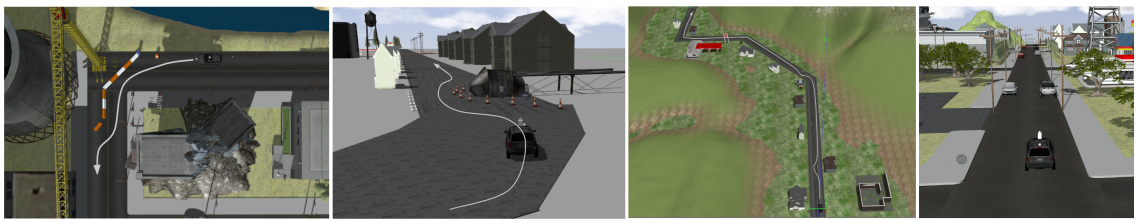


Figura 5.1 Entornos simulados de ROS.

La Figura 5.2 muestra los croquis con las formas y medidas de los segmentos de simulación en los que se realizaron las pruebas. Para validar la efectividad de cada método, se utilizaron dos escenarios cortos y dos más complejos, todos con objetos que obstruyen parcialmente el camino y uno con caminos incorrectos. Para todos los escenarios existe una trayectoria para llegar a la meta.

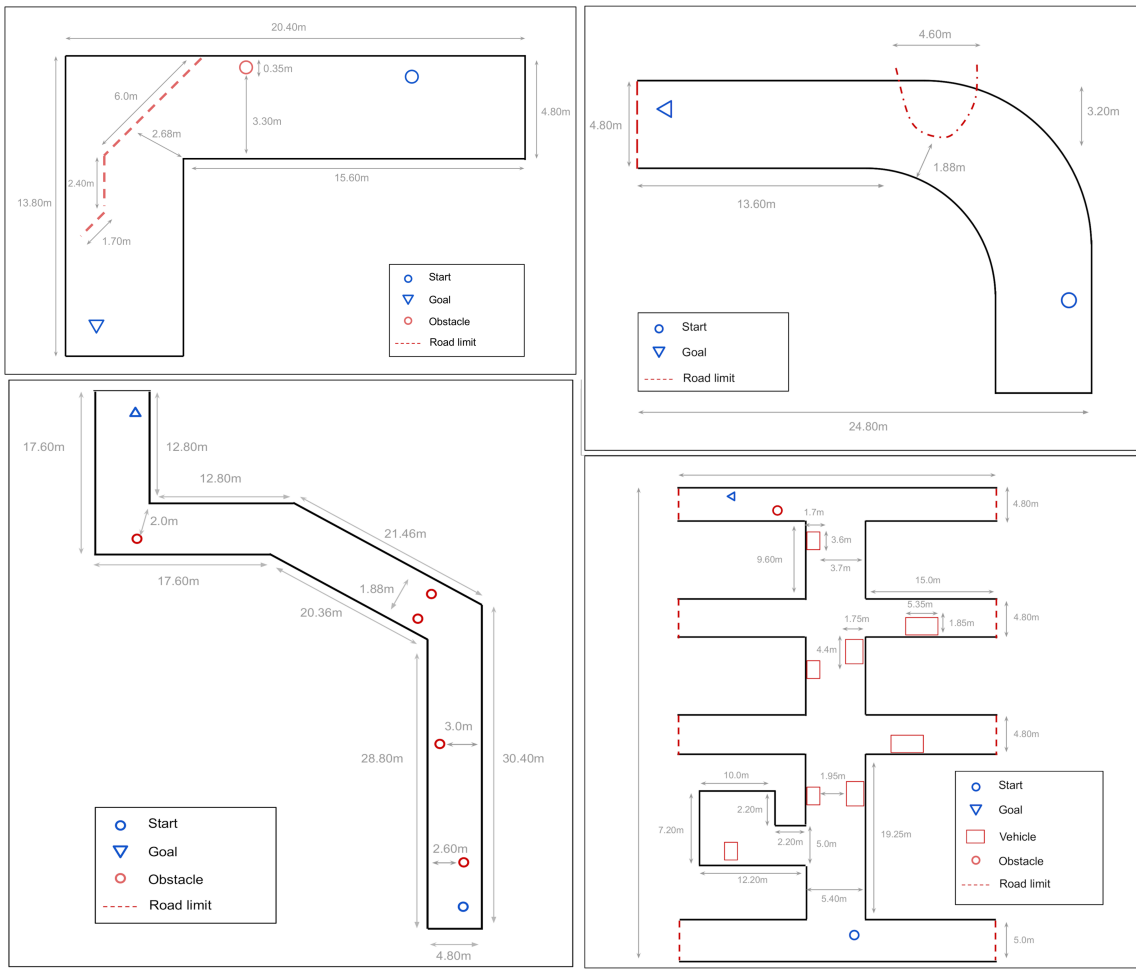


Figura 5.2 Mapas de entornos simulados a escala real para pruebas de autoconducción.

5.1.3 Experimento 2: prototipo a escala para control de dirección

En primera instancia se adquirió la base del vehículo que incluye la mecánica y los motores en funcionamiento, sus dimensiones y apariencia se muestran en la Figura 5.3.

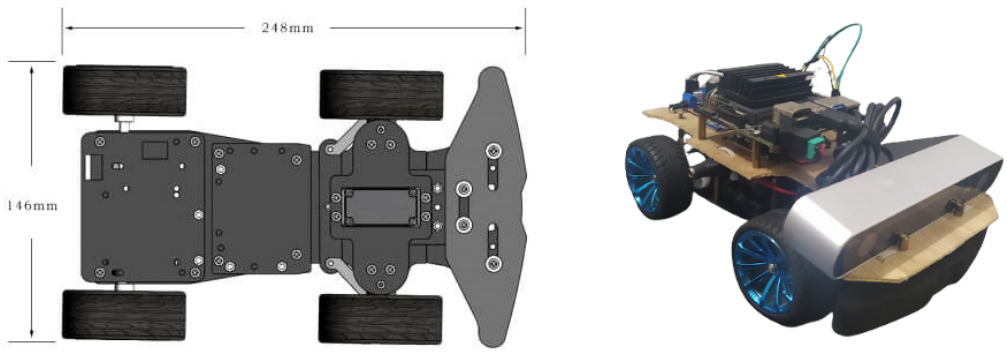


Figura 5.3 Prototipo a escala disñeado para las pruebas físicas de autoconducción.

Inspirada en el prototipo Jetracer, se utilizó una tarjeta integrada Jetson Nano Dev Kit 4G como módulo de control debido a su versatilidad en términos de programación de alto y bajo nivel, así como su capacidad informática integrada en GPU en CUDA bajo Ubuntu. Esta tarjeta incluye un procesador ARMv7 de 4 núcleos a 1,43 GHz, 4 GB de RAM de 1600 MHz, 16 GB eMMC 5.1, GPIO con soporte L2C y PWM para motores con engranajes y servo, y GPU Nvidia Maxwell 128 CUDA core. En conjunto con la tarjeta Jetson Nano se utilizaron algunos componentes electrónicos en la periferia: módulo PCA9685, conectado a los puertos GPIO dedicados a la conexión tipo L2C de la Jetson Nano. Este dispositivo se utiliza para controlar los motores transformando las señales digitales en señales PWM que dan precisión a la posición y velocidad angular del servomotor. El dispositivo puede funcionar a 3,3 V con alimentación directa del Jetson Nano, sin embargo, se requiere una alimentación externa de 5 V para mover el servomotor. Módulo L298N, sirve para ampliar el voltaje en la entrada de 3.3V del PCA9685 a la salida de 12V. Se utiliza para alimentar el motorreductor que empuja el vehículo en un canal. Regulador de voltaje CC-CC Lm2596, utilizado para convertir el voltaje de entrada de 12 V a 5 V, para alimentar el módulo PCA9685 y el Jetson Nano. Debido a las limitaciones de tamaño y potencia del prototipo, se utilizó la cámara ZED ya que permite obtener información visual, así como información espacial a partir de la generación de nubes de puntos, de manera similar al sensor LiDAR. De esta forma, se obtienen dos tipos de señales del mismo sensor.

El sistema embebido funciona de manera similar a la simulación, sin embargo el procesamiento se realiza en la tarjeta, por lo que se requirió interpretar el modelo en la versión Tensorflow Lite. Dado que la información solo se obtiene de la cámara ZED, se diseñó un sistema de adquisición en Python utilizando los drivers de la cámara. Este programa adquiere las nubes de puntos $I_d = \{M \cdot N \cdot D\}$ y crea las secuencias de imágenes en tensores de T veces en $I_t = \{M \times N \times C \times T\}$. La información de rotación y traslación también se obtiene de los controladores de la cámara. Esto genera como entrada 3 tensores 4D, 1D y 2D. La Figura 5.4 muestra el esquema del sistema implementado.

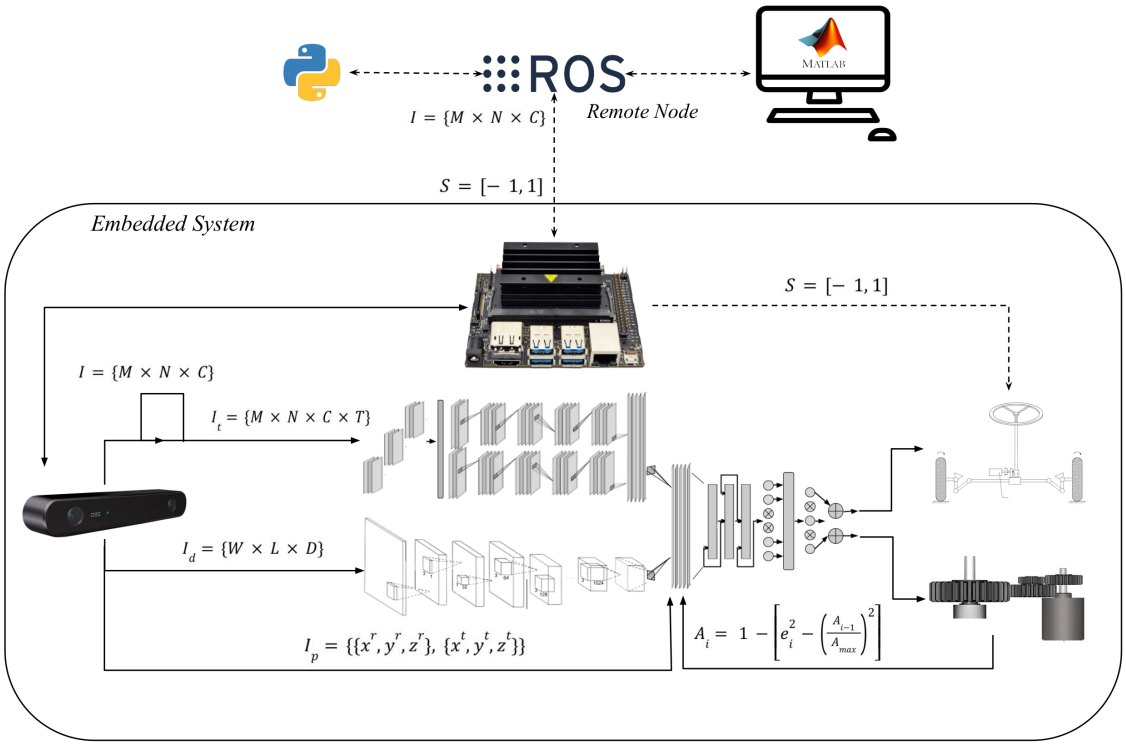


Figura 5.4 Diagrama operativo del sistema de conducción autónoma embebido e interacción remota para la evaluación de métodos de comparación [62].

La Figura 5.4 muestra la interacción de ROS con Matlab y Python fuera del sistema integrado. Estas relaciones con líneas punteadas representan las adaptaciones para poder replicar los métodos de comparación, ya que no fueron diseñados para operar en un sistema embebido. Para llevar a cabo la comunicación se utilizó un nodo ROS como intermediario en un ordenador remoto interconectado por WiFi con otro nodo de la tarjeta. El intercambio de información se realiza con nodos sincronizados para reducir la latencia, se envían imágenes de 240×134 a 10 fps desde la tarjeta a la computadora remota y devuelve un comando de control $S = [-1, 1]$.

Debido a las características del prototipo, se evaluó en escenarios viales más simples que los entornos urbanos de simulación. Los caminos diseñados están inspirados en las pistas de BlueRaven para Jetbot, adicionalmente se agregaron algunos obstáculos para validar la capacidad de evadir y corregir la trayectoria. La Figura 5.5 detalla los caminos diseñados en medidas a escala real.

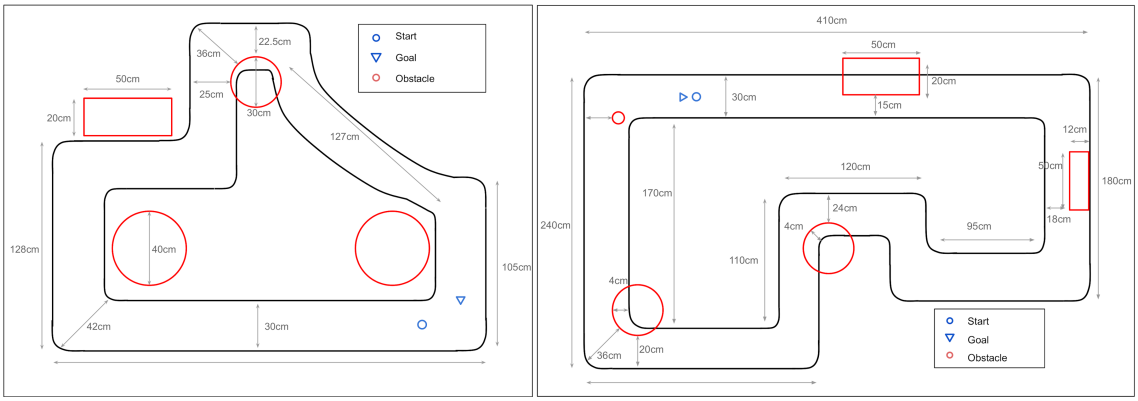


Figura 5.5 Mapas de las pistas para pruebas físicas de conducción autónoma.

5.2 Resultados

5.2.1 Autoconducción en entornos urbanos simulados

La Figura 5.2 muestra las trayectorias de conducción generadas por cada método evaluado. Se puede apreciar que en los escenarios más simples todos los métodos pueden llegar a la meta, sin embargo cabe señalar que la propuesta realiza los movimientos más suaves, acercándose a lo que hace un humano, mostrado en la verdad del terreno. En el entorno más complejo se puede observar que un método toma el camino equivocado, finalizando la evaluación del mismo. Para los casos en los que el algoritmo no puede completar la ruta, fue necesario habilitar la conducción manual para retomar la ruta, acción necesaria para calcular las métricas de Autonomía y AAT. Esta intervención también se realiza de forma intencionada cuando el desplazamiento se aleja de la verdad del suelo, obteniendo así una valoración de autonomía. Cuantitativamente, los resultados se resumen en la Tabla 5.1.

Tabla 5.1 Métricas obtenidas en la autoconducción simulada.

Method	MSE	Cosine Distance	Behavior	Path Smoothness	Autonomy	AAT
Seguimiento Reactivo	0.122	0.121, -0.223	0.01	1.26	76.1%	77.0%
Pilotnet	0.106	0.221, -0.076	0.030	0.682	86.4%	92.9%
Donkeycar	0.385	0.380, -0.510	0.236	1.826	71.3%	79.8%
MADT	0.077	0.112, -0.014	0.024	0.442	88.0%	94.7%
Proposed	0.021	0.011, -0.044	0.003	0.256	96.6%	97.4%

De acuerdo a lo mostrado en la Tabla 5.1 y con respecto a las trayectorias de la Figura 5.6, se observa que la propuesta muestra que las mejores métricas de autonomía se obtienen

por el método propuesto. Se garantiza un rango de seguridad de 96%, mientras que el peor desempeño ofrece solo 71%, siempre que se realicen intervenciones oportunas, de lo contrario se produciría una colisión. La métrica coseno distancia muestra el mayor error hacia la izquierda y hacia la derecha, para la propuesta se observa un balance de errores, mientras que MADT y Pilotnet presentan sesgos hacia la izquierda. Con base en esta experimentación, se puede ver que el método propuesto es en promedio 19% más similar a la conducción humana, respaldado por la suavidad de 0.25.

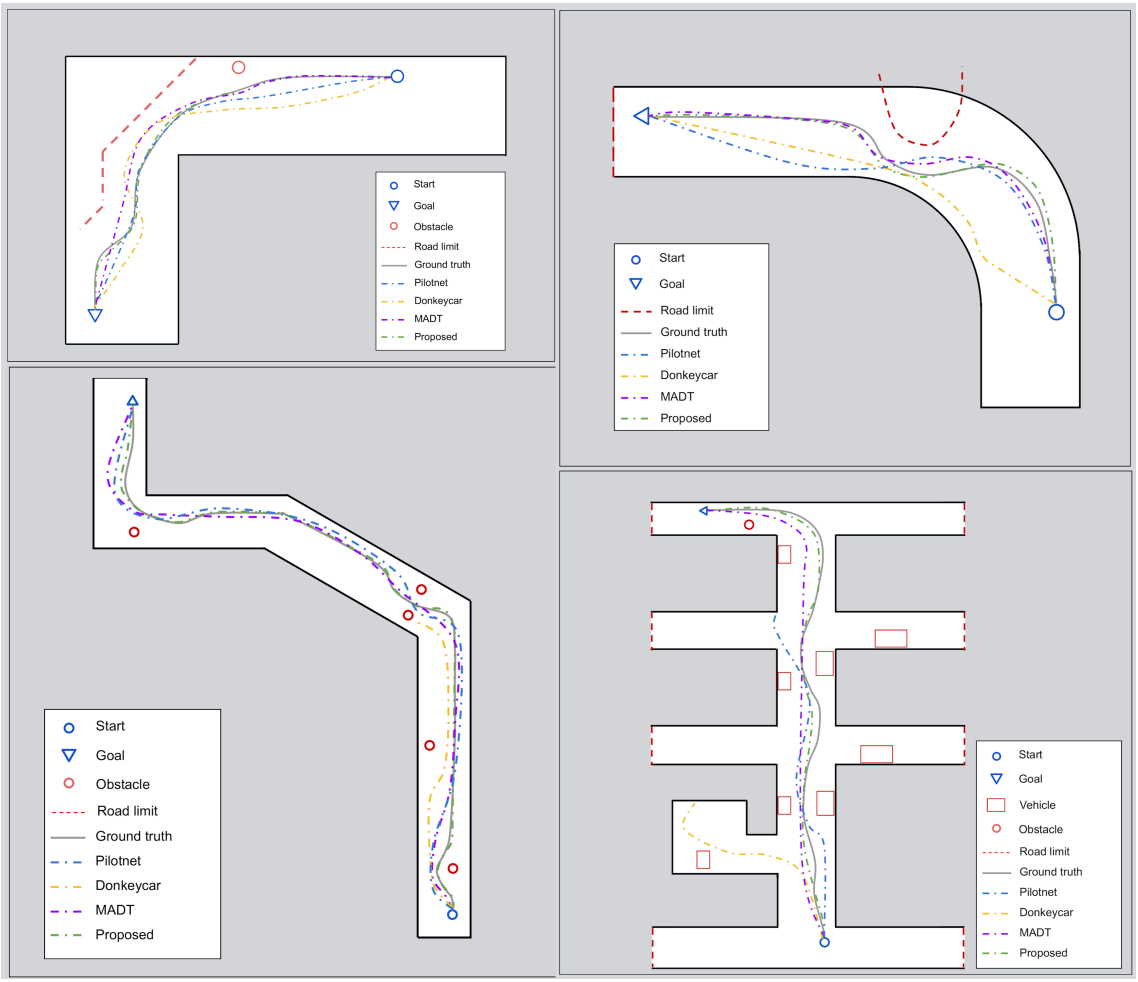


Figura 5.6 Trayectorias obtenidas durante la conducción autónoma en entornos simulados.

Como es sabido, en entornos de simulación los resultados no se ven afectados por efectos ambientales como iluminación, texturas del suelo, obstrucciones, etc. Por ello, para complementar la experimentación se diseñó un prototipo a pequeña escala que se integra en un modelo embebido en versión ligera para controlar la dirección del vehículo.

5.2.2 Autoconducción en entornos reales a escala

La Figura 5.7 muestra los caminos recorridos en las pistas de prueba. Para aumentar la experimentación, en la primera pista se modificó la ubicación de los obstáculos para validar la capacidad de evasión. En el primer escenario, todos los métodos lograron terminar el recorrido, sin embargo, al agregar los obstáculos, se observó que Donkeycar no superó el segundo obstáculo. En la pista más compleja, tanto el modelo Donkeycar como el Pilotnet no pudieron completar el recorrido completo, el último esquivó los tres obstáculos pero se salió de la carretera. En este caso, MADT también va más allá de los límites del camino, sin embargo logra retomarlo de manera autónoma. También se puede observar que tanto MADT como el método propuesto generan trayectorias muy similares en los primeros escenarios. Para realizar la comparación, se muestra un resumen de métricas en la Tabla 5.2.

Tabla 5.2 Métricas obtenidas en la autoconducción en entornos reales.

Method	**MSE**	**Cosine Distance**	**Behavior**	**Path Smoothness**	**Autonomy**	**AAT**
Seguimiento Reactivo	0.233	0.8, -1	0.421	4.6	40.9%	28.5%
Pilotnet	0.185	0.171, -0.441	0.051	1.216	73.5%	75.5%
Donkeycar	0.205	0.322, -0.571	0.087	2.948	63.8%	66.9%
MADT	0.106	0.132, -0.100	0.030	0.834	78.3%	89.8%
Proposed	0.081	0.148, -0.075	0.027	0.598	89.0%	94.4%

La Tabla 5.2 muestra el AAT de 75% para el modelo Pilotnet, esto indica que 25% del camino depende de la intervención humana, a diferencia del método propuesto que depende solo de 5.6% si se necesita una conducción perfecta . Del mismo modo, MADT obtiene un AAT de $89,8\%$ porque se desvió dos veces y debe corregirse. En este experimento la suavidad se reduce ya que se obtiene una métrica de 0.598, esto se debe a la complejidad del camino y los efectos físicos, sin embargo el método es autónomo en 89%, siendo mejor en $\approx 17.7\%$ que el promedio de los otros métodos. De estas observaciones experimentales se puede inferir que las múltiples fuentes de datos utilizadas mejoran la calidad de la conducción autónoma, así como que la capa de agregación neurofuzzy ayuda a disminuir el consumo computacional, lo que permite operar en un prototipo liviano. Además, en esta experimentación, 89% puede confiar en el método propuesto en el peor de los casos y 95% en el caso normal. Con estas observaciones específicas, se puede llegar a las conclusiones de esta investigación.

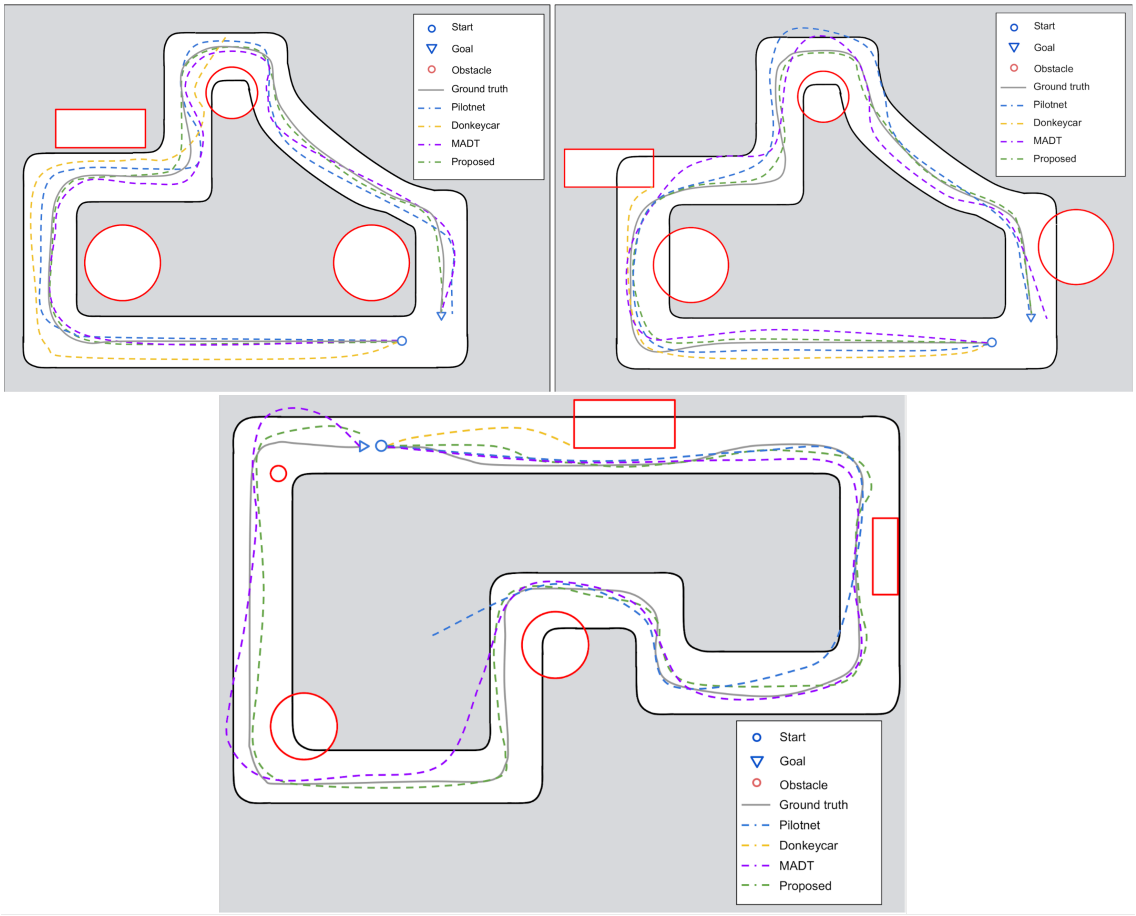


Figura 5.7 Trayectorias obtenidas durante la conducción autónoma en entornos físicos.

Aunque las métricas muestran autonomía de hasta 40.9% en el mejor de los casos, lo que representa un riesgo total de colisión, este método se muestra como el más ágil para procesar la información y generar una conducción. En la Tabla 5.3 se muestra una comparación de los tiempos de respuesta de cada método.

Tabla 5.3 Tiempos de respuesta de cada método de autoconducción

Metodo	**Resp. Simulación (ms)**	**Resp. Prototipo (ms)**	**Arquitectura de procesamiento**
Seguimiento Reactivo	15	25	CPU
Pilotnet	58	95	GPU
Donkeycar	200	355	CPU paralelo
MADT	110	420	CPU paralelo
Propuesta	31	37	GPU

La Tabla 5.3 muestra los tiempos de respuesta medios obtenidos en los entornos de prueba. El método reactivo es notablemente el más rápido, debido a que algoritmicamente es el más simple, aunque haya sido implementado de forma secuencial en CPU. En el caso de la red Pilotnet [11], su tiempo de respuesta es garantizado gracias a la implementación paralela en GPU, ya que esta red fue diseñada para la autoconducción en tiempo real, sin embargo su complejidad es alta para ser ejecutada secuencialmente en CPU. En el caso del método de Matlab MADT [66], el tiempo es de respuesta está en función a la latencia ocasionada por la comunicación entre el sistema embebido y el sistema que opera Matlab, ya que la tarjeta utilizada en el prototipo no es compatible con este. Aún así se puede observar el tiempo de respuesta de la ejecución del algoritmo de MADT en la simulación, el cual es el sistema que opera remotamente al prototipo. Finalmente se observa que el modelo propuesto, aunque no es el más rápido, disminuye considerablemente el tiempo de respuesta del modelo neuronal Pilotnet implementado bajo la misma arquitectura de procesamienta, lo que demuestra que la hipótesis planteada es correcta. Bajo las observaciones discutidas en esta sección, se llegaron a las siguientes conclusiones.

Capítulo 6

Conclusiones

En este capítulo después de haber realizado las evaluaciones correspondientes se hace un análisis de las conclusiones de la investigación a partir de los resultados obtenidos y tendiendo en cuenta los objetivos planteados. Además se describen los productos generados y las principales aportaciones. Por último se complementa con ideas y sugerencias para trabajos futuros.

6.1 Objetivos y alcances logrados

Las actividades realizadas con respecto a los objetivos y alcances planteados para el desarrollo de esta investigación se presentan en las Tablas 6.1 y 6.2.

Tabla 6.1 Objetivos cumplidos.

Objetivo	Actividad
Analizar y formular matemáticamente las capas utilizadas en las Redes Neuronales de Aprendizaje Profundo.	Se estudiaron los paradigmas neuronales y se realizó la formalización generalizada del algoritmo de aprendizaje de las capas, descrito en la Sección 2.2.1.
Analizar y formular matemáticamente medidas de agregación difusas aplicables al procesamiento de señales de dimensionalidad variable	Se estudiaron las medidas difusas y su uso en la integral difusa de Choquet para agregación de datos. En conjunto con el estudio de las redes neuronales, se redactó un reporte del estado del arte.
Formular una capa de agregación difusa compatible con las Redes Neuronales de Aprendizaje Profundo.	Se formuló matemáticamente una estructura tensorial neurodifusa, que permite la agregación de datos. Esta se describe en la Sección 4.2.
Formular una variante de algún método de entrenamiento para un modelo de Red Neuronal con capas difusas.	Se formalizó el algoritmo neurodifuso ΔW_μ, descrito en el Algoritmo 5.
Evaluar el modelo propuesto utilizando algún repositorio especializado de datos n-dimensionales	Se propuso un modelo híbrido entre autoconducción y detección de objetos, agregados mediante la capa propuesta. Se llevó a la aplicación de autoconducción a partir de la agregación de varios sensores.
Comparar con otros métodos del estado del arte usando métricas especializadas	Se comparó con métodos referentes de la literatura: Nvidia Pilotnet [11], la biblioteca de autoconducción Donkeycar [89] y *Matlab Automated Driving Toolbox* [66].

Tabla 6.2 Alcances realizados.

Alcance	Actividad
Estudiar los métodos de agregación difusa para fuentes de dimensionalidad variable.	Se realizó un análisis completo de los métodos de agregación difusa en el reporte del Proyecto de Investigación I. Se analizaron las diferentes integrales difusas Sugeno, TKS y se seleccionó por compatibilidad con la teoría propuesta a la integral de Choquet.
Estudiar los algoritmos de entrenamiento y procesamiento de las Redes Neuronales Profundas	Se redactó un análisis completo y formalización de esta teoría en los reportes de Proyecto de Investigación I y II.
Formular un método de agregación difusa interpretado como una capa neuronal con parámetros autoajustables	Esta formalización se resume de forma gráfica en la Figura 4.2 de la Sección 4.2.
Formular un algoritmo de ajuste de parámetros basado en funciones objetivo para ajustar la capa difusa	Se formuló el Algoritmo 5 de entrenamiento neurodifuso ΔW_μ.
Implementar el diseño, entrenamiento y prueba de los modelos neuronales en un lenguaje de bajo nivel paralelizado (GPU).	Se redactó un artículo de investigación sobre este punto, presentado en el *2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control* [61]
Evaluar los resultados en términos de calidad de fusión, clasificación y operación del modelo neuronal mediante métricas especializadas del Estado del Arte	Se redactó el artículo de investigación publicado en la revista *Electronics* con factor de impacto de 2.9 [62].
Comparar los resultados con distintas técnicas de fusión y clasificación, por separado y como un conjunto de procesamiento	Se realizó la comparación entre las integrales difusas para fusión, como conjunto se comparó en términos de autoconducción de vehículos, mostrado en la Sección 5.

6.2 Resultados de la investigación

6.2.1 Productos

Durante el desarrollo de esta investigación se obtuvieron los siguientes productos:

1. Reporte del estado del arte. Este documento se divide en la investigación sobre la teoría y aplicaciones de las medidas μ de agregación difusa, integrales difusas, modelos neuronales, sistemas de autoconducción y estado de la práctica con documentos de patentes.

2. Formulación e implementación de la capa de agregación neurodifusa: a bajo nivel (ANSI-C y CUDA) la creación de estructuras tensoriales, a alto nivel (Tensorflow 2) el resto de los componentes neuronales.

3. Formulación e implementación del modelo de autoconducción multisensorial: creado a partir de plantillas personalizadas de Tensorflow.

4. Artículo publicado en el congreso *2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)* [61], Figura A.1.

5. Artículo aceptado en la revista indizada *Journal of Applied Research and Technology* (Q2 CONACYT), Figura A.2.

6. Artículo publicado en la revista *Electronics* de la editorial MDPI (IF 2.9) en la edición especial *"Autonomous Vehicles Technological Trends"*, Figuras A.3, A.4.

6.2.2 Aportaciones

Las aportaciones obtenidas con esta investigación son:

- Formulación matemática de la capa de agregación neurodifusa.
- Creación de un modelo de autoconducción multisensorial a partir de la agregación neurodifusa de las señales de dos modelos neuronales profundos.
- Sistema de autoconducción en entornos urbanos simulados en ROS.
- Sistema de autoconducción embebida, con capacidad de temeraria compatible con ROS y Matlab.

6.3 Conclusiones

En este estudio, se propuso una capa neuronal de agregación neurodifusa para fusionar datos procesados por un modelo neuronal y reducir su dimensionalidad de tensor, con el fin de componer nuevas características obtenidas a partir de múltiples entradas de datos. La capa propuesta se implementó para diseñar una arquitectura compuesta por un modelo autónomo con series temporales y otro para detección de objetos, fusionando ambas fuentes con información de traslación y rotación. La aplicación fue desarrollada en ambientes de simulación ROS y en un prototipo a escala.

Como conclusión general del trabajo, las principales contribuciones de esta investigación son: integración de modelos en un solo proceso, agregación de múltiples fuentes de datos, un modelo de conducción autónoma multisensorial, se obtuvo una autonomía superior al 95% y una similitud de $\approx 98\%$ con el conducta de conducción humana. La La experimentación mostró que la integración de múltiples fuentes de datos mejora la fluidez de conducción en 9% mientras es confiable 89% del tiempo.

Como conclusión después de experimentar con el método clásico de seguimiento de líneas se observó que es hasta un 3.8 veces más rápido que una implementación en GPU de una red neuronal. Haciendo un análisis del método, tiene una complejidad $O(n^2+n)$, en comparación que un modelo como Pilotnet de complejidad polinomial, aun cuando se considera un modelo de complejidad media entre el paradigma del aprendizaje profundo. Por otro lado, aunque el modelo propuesto no es el más rápido, es 2.5 veces más rápido que el modelo Pilotnet, el cual es la base de la mitad del modelo propuesto.

6.3.1 Trabajo futuro

Los objetivos y alcances del proyecto de investigación se han concluido en su totalidad. Como trabajo futuro, en próximos proyectos se plantea experimentar con más sensores y realizar tareas como detección de señalamientos de tráfico, peatones, así como integrar la capacidad de planificación y seguimiento de rutas. Como próxima investigación, se podría extender a un prototipo que permita a un usuario simular la conducción intensivamente y recibir la asistencia de conducción desde el punto de autoconducción.

Referencias

[1] Agrawal, U., Wagner, C., Garibaldi, J. M., and Soria, D. (2019). Fuzzy integral driven ensemble classification using a priori fuzzy measures. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE.

[2] Ahlander, K. (2002). Einstein summation for multidimensional arrays. *Computers & Mathematics with Applications*, 44(8-9):1007–1017.

[3] Alatise, M. B. and Hancke, G. P. (2020). A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846.

[4] Andersen, J. J., Erenel, D., Lyle, R. O., and Yee, C. (2019). Sensor fusion model to enhance machine conversational awareness. US Patent App. 15/806,438.

[5] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[6] Beke, A. and Kumbasar, T. (2019). Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks. *Engineering Applications of Artificial Intelligence*, 85:372–384.

[7] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[8] Benseddik, H. E., Morbidi, F., and Caron, G. (2020). Panoramis: An ultra-wide field of view image dataset for vision-based robot-motion estimation. *The International Journal of Robotics Research*, pages 1–14.

[9] Bhowal, P., Sen, S., Velasquez, J. D., and Sarkar, R. (2022). Fuzzy ensemble of deep learning models using choquet fuzzy integral, coalition game and information theory for breast cancer histology classification. *Expert Systems with Applications*, 190:116167.

[10] Blanco-Claraco, J.-L., Moreno-Dueñas, F.-Á., and González-Jiménez, J. (2014). The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario. *The International Journal of Robotics Research*, 33(2):207–214.

[11] Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., and Muller, U. (2017). Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*.

[12] Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168.

[13] Bourhim, A., Mashreghi, J., Oubbi, L., and Abdelali, Z. (2020). *Linear and Multilinear Algebra and Function Spaces*, volume 750. American Mathematical Soc.

[14] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163.

[15] Carlevaris-Bianco, N., Ushani, A. K., and Eustice, R. M. (2016). University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035.

[16] Castellano Quero, M., Fernández-Madrigal, J. A., Garcia-Cerezo, A. J., et al. (2018). Hacia la diagnosis y fusión de sensores robóticos a bajo nivel mediante inferencia en redes bayesianas.

[17] Castro, A. A. R. (2020). Sistema embebido para asistencia de conducción basado en lógica difusa tipo-2. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

[18] Cervantes, A. P. (2023). Detección de obstáculos y planos durante el desplazamiento vehicular. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

[19] Chen, C., Rosa, S., Miao, Y., Lu, C. X., Wu, W., Markham, A., and Trigoni, N. (2019). Selective sensor fusion for neural visual-inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10542–10551.

[20] Chen, J., Qiang, H., Wu, J., Xu, G., and Wang, Z. (2021). Navigation path extraction for greenhouse cucumber-picking robots using the prediction-point hough transform. *Computers and Electronics in Agriculture*, 180:105911.

[21] Chen, S., Zhang, S., Shang, J., Chen, B., and Zheng, N. (2017a). Brain-inspired cognitive model with attention for self-driving cars. *IEEE Transactions on Cognitive and Developmental Systems*, 11(1):13–25.

[22] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017b). Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915.

[23] Chen, Y. and Blum, R. S. (2009). A new automated quality assessment algorithm for image fusion. *Image and vision computing*, 27(10):1421–1432.

[24] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[25] Chollet, F. (2017). Deep learning with python, vol. 1. *Greenwich, CT: Manning Publications CO*.

[26] Cruz-Hernandez, J. M., Grant, D. A., and Ullrich, C. J. (2020). Systems and methods for multi-rate control of haptic effects with sensor fusion. US Patent 10,649,532.

[27] Cui, G., Feng, H., Xu, Z., Li, Q., and Chen, Y. (2015). Detail preserved fusion of visible and infrared images using regional saliency extraction and multi-scale image decomposition. *Optics Communications*, 341:199–209.

[28] De Silva, V., Roche, J., and Kondoz, A. (2018). Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 18(8):2730.

[29] Dozat, T. (2016). Incorporating nesterov momentum into adam.(2016). *Dostupné z: http://cs229. stanford. edu/proj2015/054_report. pdf*.

[30] Drewek-Ossowicka, A., Pietrołaj, M., and Rumiński, J. (2021). A survey of neural networks usage for intrusion detection systems. *Journal of Ambient Intelligence and Humanized Computing*, 12(1):497–514.

[31] Du, X., Ang, M. H., and Rus, D. (2017). Car detection for autonomous vehicle: Lidar and vision fusion approach through deep learning framework. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 749–754. IEEE.

[32] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

[33] Ellmauthaler, A., Pagliari, C. L., da Silva, E. A., Gois, J. N., and Neves, S. R. (2019). A visible-light and infrared video database for performance evaluation of video/image fusion methods. *Multidimensional Systems and Signal Processing*, 30(1):119–143.

[34] Emadi, M. and Rahgozar, M. (2020). Twitter sentiment analysis using fuzzy integral classifier fusion. *Journal of Information Science*, 46(2):226–242.

[35] Estrada-Trejo, A. C., Ramos-Silva, I., Sepúlveda-Cervantes, G., Portilla-Flores, E. A., and Vega-Alvarado, E. (2019). Predicción de trayectorias usando el filtro de kalman. *Research in Computing Science*, 148:307–316.

[36] Etingof, P., Gelaki, S., Nikshych, D., and Ostrik, V. (2016). *Tensor categories*, volume 205. American Mathematical Soc.

[37] Gao, B. and Pavel, L. (2017). On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*.

[38] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[39] Gillian, N. E., Schwesig, C. C., Lien, J., Amihood, P. M., and Poupyrev, I. (2019). Radar-enabled sensor fusion. US Patent 10,401,490.

[40] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[41] Grabisch, M., Sugeno, M., and Murofushi, T. (2010). *Fuzzy measures and integrals: theory and applications*. Heidelberg: Physica, 2000.

[42] Gross, R., Matthews, I., Cohn, J., Kanade, T., and Baker, S. (2010). Multi-pie. *Image and Vision Computing*, 28(5):807–813.

[43] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

[44] Honda, A. and James, S. (2020). Parameter learning and applications of the inclusion-exclusion integral for data fusion and analysis. *Information Fusion*, 56:28–38.

[45] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[46] Huang, Y., Moore, J. E., et al. (2021). Neural network representation of tensor network and chiral states. *Physical Review Letters*, 127(17):170601.

[47] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[48] Islam, M. A., Anderson, D. T., Pinar, A., Havens, T. C., Scott, G., and Keller, J. M. (2019). Enabling explainable fusion in deep learning with fuzzy integral neural networks. *IEEE Transactions on Fuzzy Systems*.

[49] Jagalingam, P. and Hegde, A. V. (2015). A review of quality metrics for fused image. *Aquatic Procedia*, 4(Icwrcoe):133–142.

[50] Jeong, J., Cho, Y., Shin, Y.-S., Roh, H., and Kim, A. (2019). Complex urban dataset with multi-level sensors from highly diverse urban environments. *The International Journal of Robotics Research*, 38(6):642–657.

[51] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

[52] Kanetkar, Y. P. (2016). *Let us C*. BPB publications.

[53] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[54] Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., and Čehovin, L. (2016). A novel performance evaluation methodology for single-target trackers. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2137–2155.

[55] Li, J. (2015). Fusion of lidar 3d points cloud with 2d digital camera image. *Oakland University: Rochester, MI, USA*.

[56] Liang, M., Yang, B., Wang, S., and Urtasun, R. (2018). Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656.

[57] Liggins, M., Hall, D., and Llinas, J. (2017). *Handbook of multisensor data fusion: theory and practice*. CRC press.

[58] Lin, C.-J., Lin, C.-H., Wang, S.-H., and Wu, C.-H. (2019). Multiple convolutional neural networks fusion using improved fuzzy integral for facial emotion recognition. *Applied Sciences*, 9(13):2593.

[59] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. (2010). The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pages 94–101. IEEE.

[60] Luna, A. (2020). Conducción autónoma de un vehículo simulado mediante un modelo de red neuronal convolucional recurrente. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

[61] Luna-Alvarez, A., Mújica-Vargas, D., Matuz-Cruz, M., Kinani, J. M. V., and Ramos-Díaz, E. (2020). Self-driving through a time-distributed convolutional recurrent neural network. In *2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–6. IEEE.

[62] Luna-Álvarez, A., Mújica-Vargas, D., Rendón-Castro, A., Matuz-Cruz, M., and Kinani, J. M. V. (2023). Neurofuzzy data aggregation in a multisensory system for self-driving car steering. *Electronics*, 12(2):314.

[63] Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15.

[64] Mahmood, A., Baig, A., and Ahsan, Q. (2012). Real time localization of mobile robotic platform via fusion of inertial and visual navigation system. In *2012 International Conference of Robotics and Artificial Intelligence*, pages 40–44. IEEE.

[65] Majdik, A. L., Till, C., and Scaramuzza, D. (2017). The zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, 36(3):269–273.

[66] MathWorks (2022). Automated driving toolbox.

[67] Michelucci, U. (2018). *Applied deep learning: a case-based approach to understanding deep neural networks*. Apress.

[68] Morales, M. (2018). Segmentación no paramétrica de tejidos cerebrales mediante unaarquitectura paralela de redes neuronales convolucionales. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

[69] Mu, Z. and Zeng, S. (2019). Some novel intuitionistic fuzzy information fusion methods in decision making with interaction among attributes. *Soft Computing*, 23(20):10439–10448.

[70] Mújica-Vargas, D., Luna-Álvarez, A., de Jesús Rubio, J., and Carvajal-Gámez, B. (2020). Noise gradient strategy for an enhanced hybrid convolutional-recurrent deep network to control a self-driving vehicle. *Applied Soft Computing*, page 106258.

[71] Mukunoki, D. and Ogita, T. (2020). Performance and energy consumption of accurate and mixed-precision linear algebra kernels on gpus. *Journal of Computational and Applied Mathematics*, 372:112701.

[72] Navarrete, L. (2019). Sistema de navegación inercial asistido por visión para robots móviles terrestres. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

[73] Paszke, A., Gross, S., Chintala, S., and Chanan, G. (2017). Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 6:3.

[74] Pfrommer, B., Sanket, N., Daniilidis, K., and Cleveland, J. (2017). Penncosyvio: A challenging visual inertial odometry benchmark. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3847–3854. IEEE.

[75] R, R. and R, P. (2018). Hybrid multimodality medical image fusion technique for feature enhancement in medical diagnosis. *International Journal of Engineering Science Invention*.

[76] Raaj, Y., John, A., and Jin, T. (2016). 3d object localization using forward looking sonar (fls) and optical camera via particle filter based calibration and fusion. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–10. IEEE.

[77] Ran, S.-J., Tirrito, E., Peng, C., Chen, X., Tagliacozzo, L., Su, G., and Lewenstein, M. (2020). Tensor network contraction and multi-linear algebra. In *Tensor Network Contractions*, pages 99–129. Springer.

[78] Rivera, F. J. V. (2022). Navegación de un robot omnidireccional basada en lógica difusa tipo-2. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

[79] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[80] Saleh, E., Valls, A., Moreno, A., Romero-Aroca, P., Torra, V., and Bustince, H. (2018). Learning fuzzy measures for aggregation in fuzzy rule-based models. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 114–127. Springer.

[81] Scavezze, M., Tomlin, A., Cai, R., and Li, Z. (2019). Optimized object scanning using sensor fusion. US Patent 10,257,505.

[82] Scherer, S., Yu, S., and Nuske, S. (2019). State estimation for aerial vehicles using multi-sensor fusion. US Patent 10,295,365.

[83] Schubert, D., Goll, T., Demmel, N., Usenko, V., Stückler, J., and Cremers, D. (2018). The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687. IEEE.

[84] Shah, A. A., Chowdhry, B. S., Memon, T. D., Kalwar, I. H., and Ware, A. (2020). Real time identification of railway track surface faults using canny edge detector and 2d discrete wavelet transform. *Annals of Emerging Technologies in Computing (AETiC)*, 4(2).

[85] Shakeel, P. M., Arunkumar, N., and Abdulhay, E. (2018). Automated multimodal background detection and shadow removal process using robust principal fuzzy gradient partial equation methods in intelligent transportation systems. *International Journal of Heavy Vehicle Systems*, 25(3-4):271–285.

[86] Sohn, J. B. (2019). Nine-axis quaternion sensor fusion using modified kalman filter. US Patent 10,274,318.

[87] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[88] Stanley, J. and Steinhardt, B. (2014). Bigger monster, weaker chains: The growth of an american surveillance society. In *Ethics and Emerging Technologies*, pages 269–284. Springer.

[89] Subedi, S. (2020). Ai in robotics: Implementing donkey car.

[90] Sugeno, M. (1974). Theory of fuzzy integrals and its applications. *Doct. Thesis, Tokyo Institute of technology*.

[91] Sun, Z., Tian, L., Du, Q., and Bhutto, J. A. (2022). Sample hardness guided softmax loss for face recognition. *Applied Intelligence*, pages 1–16.

[92] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*.

[93] Urban, S. and Jutzi, B. (2017). Lafida—a laserscanner multi-fisheye camera dataset. *Journal of Imaging*, 3(1):5.

[94] Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., and Chervyakov, N. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*.

[95] Wei, G. (2017). Picture fuzzy aggregation operators and their application to multiple attribute decision making. *Journal of Intelligent & Fuzzy Systems*, 33(2):713–724.

[96] Wu, Y., Lim, J., and Yang, M.-H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848.

[97] Xilot, J. E. H., Castillo, G. M., Guerrero, E. B., and Godoy, C. M. (2018). Hacia un mecanismo de razonamiento para la inferencia de contextos basada en la fusión de flujos de datos (towards a reasoning mechanism for the contexts inference based on the fusion of data flows). *Pistas Educativas*, 40(130).

[98] Zhang, X., Ye, P., and Xiao, G. (2020). Vifb: A visible and infrared image fusion benchmark. *arXiv preprint arXiv:2002.03322*.

[99] Zhang, Y., Ishibuchi, H., and Wang, S. (2017). Deep takagi–sugeno–kang fuzzy classifier with shared linguistic fuzzy rules. *IEEE Transactions on Fuzzy Systems*, 26(3):1535–1549.

[100] Zhang, Z., Rebecq, H., Forster, C., and Scaramuzza, D. (2016). Benefit of large field-of-view cameras for visual odometry. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808. IEEE.

Apéndice A

Producción

Figura A.1 Constancia de autor del artículo publicado en el congreso internacional CCE 2020.

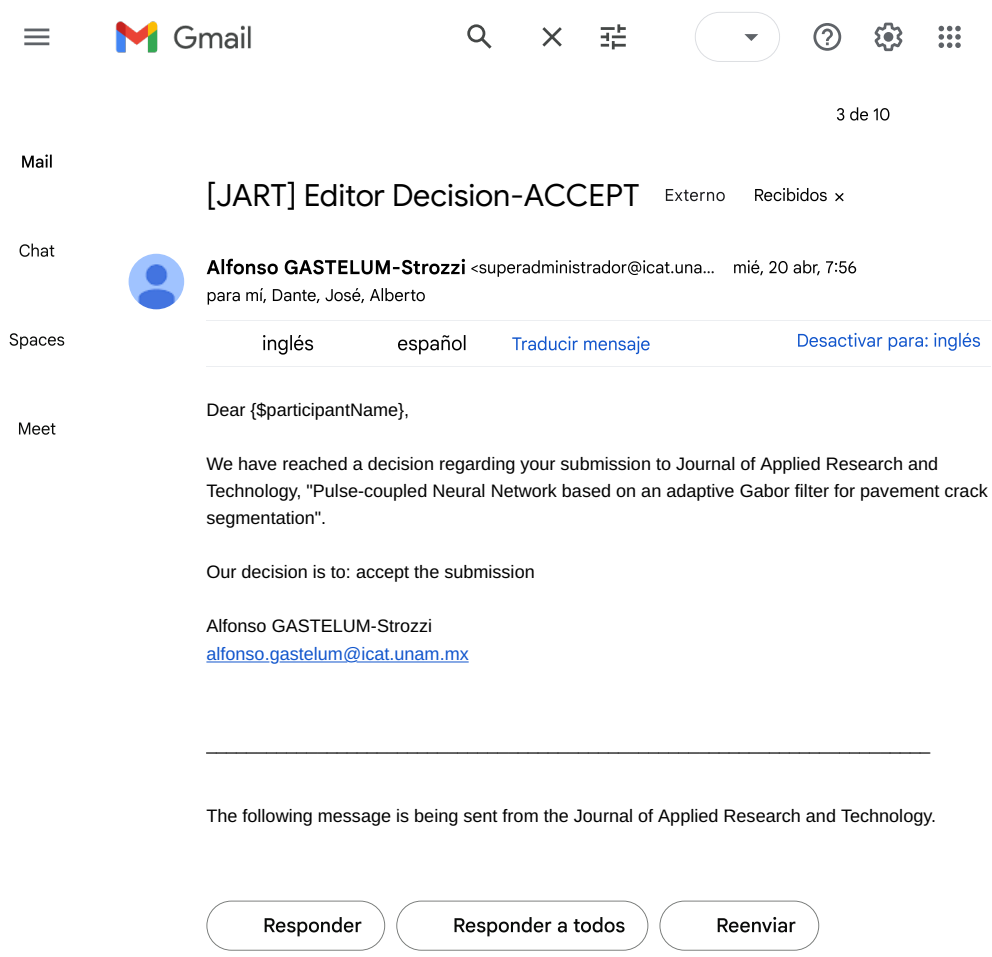


Figura A.2 Notificación de aceptación del artículo en la revista JART el 20 de abril de 2022.

MDPI

Article

Neurofuzzy Data Aggregation in a Multisensory System for Self-driving Car Steering

Antonio Luna-Álvarez [1], Dante Mújica-Vargas [1,*], Arturo Rendón-Castro [1], Manuel Matuz-Cruz [2], Jean Marie Vianney-Kinani [3]

[1] Department of Computer Science, Tecnológico Nacional de México/CENIDET, Interior Internado Palmira S/N, Palmira, Cuernavaca 62490, México
[2] Departamento de Sistemas Computacionales, Tecnológico Nacional de México/ITTapachula, Tapachula Chiapas 30700, México
[3] Unidad Profesional Interdisciplinaria de Ingeniería campus Hidalgo, Instituto Politécnico Nacional, Pachuca, Hidalgo. México
* Correspondence: dante.mv@cenidet.tecnm.mx

Abstract: This article presents a deep learning neural layer formulation for data fusion using fuzzy aggregation, to control a self-driving vehicle steering. Unlike methods that perform data fusion and further learning as separate processes, this approach integrates aggregation using fuzzy μ measures as fuzzy synaptic weights, hidden state using Choquet fuzzy integral, and an fuzzy backpropagation algorithm, creating a neurofuzzy data processing from different sources. The performance is evaluated by metrics oriented to fusion quality, precision and driving autonomy. The experimental results demonstrate the improvement in the vehicle autonomy, the driving quality and the computational load reduction.

Keywords: Deep Learning; Data fusion; Fuzzy Measure; Neurofuzzy Data Processing; Fuzzy Backpropagation; Self-driving; Vehicle Autonomy

Citation: Luna-Álvarez, A.; Mújica-Vargas, D.; Rendón-Castro, A.; M. Matuz-Cruz; Vianney-Kinani, J. Neurofuzzy Data Aggregation in a Multisensory System for Self-driving Car Steering. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

In systems, the use of multiple data sources allows an actuator to have a broad view of the environment it is seeking to control, allowing it to perform this task with greater precision and to tolerate anomalies in data from some source [1]. This method is used in physical and robotic systems to widen the field of perception and avoid blind spots, while the redundancy of the data allows the controller system to detect anomalies and filter them [2,3]. Mobile robots use this method to perceive their environment from different sensors that provide spatial, inertial and visual information [4,5], this is most notable in the Autonomous Vehicles application [6,7]. The research area for vehicles control is divided into sub areas such as: environment perception, object and vehicles recognition, behaviors, planning and route selection, lane maintenance, signal detection, steering and speed control, among others [8]. This research focuses on the sensor data fusion for steering control.

To control the vehicle direction, strategies based on reinforcement learning [9,10], fuzzy inference [11,12], predictive models [13–16] have been proposed, however the majority focuses on deep learning models. For proposals based on deep learning, the control strategy through vision and detection is the most used [17–19], as well as recurrent networks for visual-temporal relationship [20,21] and to improve performance, multisensory systems [22–24]. One of the approaches to process multiple data sources is to create parallel architectures that process each source separately and integrate the outputs [25–27], on the other hand a priori data fusion methods are proposed [28–30], also deep learning fusion methods [31,32].

Although the aforementioned works present good results, they have some weaknesses. In parallel models, the computational cost increases according to the increase in sensors to be processed, making this option viable only in simulation, since it is too much load for

Figura A.3 Primera página del artículo publicado en la revista *Electronics*.

electronics

an Open Access Journal by MDPI

CITESCORE 3.7

IMPACT FACTOR 2.690

CERTIFICATE OF ACCEPTANCE

Certificate of acceptance for the manuscript (**electronics-2126002**) titled:

Neurofuzzy Data Aggregation in a Multisensory System for Self-driving Car Steering

Authored by:

Antonio Luna-Alvarez; Dante Mújica-Vargas; Arturo Rendon-Castro; Manuel Matuz-Cruz; Jean-Marie Vianney-Kinani

has been accepted in *Electronics* (ISSN 2079-9292) on 03 January 2023

MDPI Academic Open Access Publishing since 1996

Basel, January 2023

Figura A.4 Certificado de aceptación del artículo *Neurofuzzy Data Aggregation in a Multisensory System for Self-driving Car Steering* a la revista *Electronics*.

Apéndice B

Retribución social

Figura B.1 Constancia por impartir el taller "REDES NEURONALES PARA RECONOCIMIENTO VISUAL CON PYTHON YTENSORFLOW", en el marco del 10º Congreso Internacional de Computación 2020.

Figura B.2 Constancia por impartir el taller "Sistemas Híbridos de Visión Artificial y Programación Heterogénea",en el 11° Congreso Internacional de Computación.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL INSTITUTO TECNOLÓGICO DE TAPACHULA

OTORGA EL PRESENTE

RECONOCIMIENTO

A

M. C. JESÚS ANTONIO LUNA ÁLVAREZ

Por haber participado en el ***"Coloquio Virtual: La IA en el Entorno Universitario"***, con el tema ***"Redes Neuronales y Deep Learning"*** realizado el 07 de julio de 2020.

TAPACHULA CHIAPAS, A 24 JULIO DE 2020.

ESTELA RIVERA LÓPEZ
DIRECTORA

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE TAPACHULA
DIRECCIÓN

Folio: ITTAP – 20 – D – 0672

Figura B.3 Constancia por haber participado en el "Coloquio Virtual: La IA en el Entorno Universitario", con el tema “Redes Neuronales y Deep Learning”.

La Sociedad Mexicana de Ciencia de la Computación A.C.
extiende este Certificado a nombre de:

Antonio Luna Alvarez

Como Socio Estudiante no. 105 con validez durante el periodo Agosto 2021-Agosto 2022

DRA. KARINA MARIELA FIGUEROA MORA
Presidenta de la SMCC 2021- 2023

Figura B.4 Certificado como socio estudiante de la Sociedad Mexicana de Ciencia de la Computación A.C.

Figura B.5 Reconocimiento Por haber impartido la conferencia "DESARROLLO DE UN SISTEMA DE CONDUCCIÓN AUTÓNOMA BASADO EN UNA RED NEURONAL CON PROGRAMACIÓN HETEROGÉNEA", en el Instituto Tecnológico de Cuautla