SEP **TNM**

INSTITUTO TECNOLÓGICO DE TIJUANA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MODELO DE APRENDIZAJE NEURONAL DIFUSO PROFUNDO APLICADO A PROBLEMAS DE DETECCIÓN Y CLASIFICACIÓN

TRABAJO DE TESIS

PRESENTADO POR
ING. CESAR ALAN TORRES QUIROZ

PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

DIRECTOR DE TESIS
DRA. CLAUDIA IBETH GONZÁLEZ BERRELLEZA

CO-DIRECTOR DE TESIS
DRA. GABRIELA ELIZABETH MARTÍNEZ MENDIVIL

TIJUANA, B. C. MÉXICO, Marzo 2023

Tijuana, Baja California, 09/marzo/2023
OFICIO No. 024/DEPI/2023
Asunto: Autorización de Impresión de Tesis

MARÍA MAGDALENA SERRANO ORTEGA
JEFA DEL DEPARTAMENTO DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, "Modelo de aprendizaje neuronal difuso profundo aplicado a problemas de detección y clasificación" Presentado por C. **Cesar Alan Torres Quiroz,** alumno de la Maestría en Ciencias de la Computación con numero de control **M21210008;** informo a usted que a solicitud del comité de tutorial, tengo a bien **Autorizar la impresión de Tesis,** atendiendo las disposiciones de los Lineamientos para la Operación de Estudios de Posgrado del Tecnológico Nacional de México.

Sin más por el momento le envió un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Por una Juventud Integrada al Desarrollo de México®

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE TIJUANA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

GUADALUPE HERNÁNDEZ ESCOBEDO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
GHE/lap

Tijuana, B.C., 01 de Marzo de 2023.
Asunto: Se autoriza impresión de Trabajo de Tesis

C. Dra. María Magdalena Serrano Ortega
Jefe del Depto. de Servicios Escolares
Presente.

En lo referente al trabajo de tesis escrito, con título " **MODELO DE APRENDIZAJE NEURONAL DIFUSO PROFUNDO APLICADO A PROBLEMAS DE DETECCIÓN Y CLASIFICACIÓN** ", presentado por el ***C. CESAR ALAN TORRES QUIROZ*** alumno de la Maestría en Ciencias de la Computación con número de control **M21210008**, informamos a usted que después de una minuciosa revisión, de acuerdo con lo establecido en el reglamento vigente para este caso, nuestro dictamen es: Se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de maestría y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo.

ATENTAMENTE

DRA. CLAUDIA IBETH GONZALEZ BERRELLEZA
PRESIDENTE

DRA. GABRIELA ELIZABETH MARTINEZ MENDIVIL
SECRETARIO

DR. FEVRIER ADOLFO VALDEZ ACOSTA
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf

Tijuana, B.C., 01 de Marzo de 2023.
Asunto: Se autoriza impresión de Trabajo de Tesis

C. Dr. Guadalupe Hernández Escobedo
Jefe de División de Estudios de Posgrado e Inv.
Presente.

En lo referente al trabajo de tesis escrito, con título " **MODELO DE APRENDIZAJE NEURONAL DIFUSO PROFUNDO APLICADO A PROBLEMAS DE DETECCIÓN Y CLASIFICACIÓN** ", presentado por el ***C. CESAR ALAN TORRES QUIROZ*** alumno de la Maestría en Ciencias de la Computación con número de control **M21210008**, informamos a usted que se autoriza el escrito de tesis y se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de maestría y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo y a presentar su examen de grado, ya que cumple con todos los requisitos.

ATENTAMENTE

DRA. CLAUDIA IBETH GONZALEZ BERRELLEZA
PRESIDENTE

DRA. GABRIELA ELIZABETH MARTINEZ MENDIVIL
SECRETARIO

DR. FEVRIER ADOLFO VALDEZ ACOSTA
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf

CARTA DECLARACIÓN DE PROPIEDAD INTELECTUAL

Tijuana, BC a 1 de febrero del 2023

Yo Cesar Alan Torres Quiroz reconozco que el Trabajo de Tesis de Maestría que realice durante mis estudios en la Maestría en Ciencias de la Computación del Instituto Tecnológico de Tijuana fue parte del Proyecto de Investigación Titulado**: Modelo de inteligencia artificial basado en redes neuronales profundas y lógica difusa aplicado a problemas de detección y clasificación No14056.22-P.** que desarrolla mi directora de tesis el **Dra. Claudia Ibeth González Berrelleza.** -y del cual es responsable del proyecto de investigación. Por esta razón, los métodos, modelos, algoritmos, y software realizados, así como datos y resultados obtenidos durante el desarrollo de mi tesis de maestría son propiedad intelectual de mi Director de Tesis, del Tecnológico Nacional de México, Instituto Tecnológico de Tijuana y del Conacyt, y No podré utilizarlos por mi cuenta durante, Ni después de terminar mi beca o estudios, excepto a solicitud escrita para poder utilizarlos bajo una colaboración directa con mi directora de tesis la cual es responsable del proyecto de investigación. Por tanto, estoy de acuerdo en que No podre utilizar ni tomar modelos, ni datos utilizados en este proyecto de investigación y en el desarrollo de tesis para: presentaciones, publicaciones ni desarrollo de mi propia investigación que pudiera desarrollar una vez concluidos mis estudios**.**

Atentamente

Cesar Alan Torres Quiroz

Cesar Torres.

Estudiante de la Maestría en Ciencias de la Computación

DECLARACIÓN DE ORIGINALIDAD

Tijuana, BC., 1 de febrero de 2023,

Yo, **Cesar Alan Torres Quiroz** estudiante de la Maestría en Ciencias de la Computación, en mi calidad de autor manifiesto que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patente y similaridad. Por lo tanto, la obra realizada es de mi exclusiva autoría y no infringí en copiar el texto o imágenes, de fuentes de información por lo cual soy responsable del escrito que aquí se presenta.

Así mismo, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

En caso de presentarse cualquier reclamación o acción por parte de terceros en cuanto a los derechos de autor sobre la obra en cuestión, acepto toda la responsabilidad de tal infracción y relevo de esta a mi director de tesis, así como al Tecnológico Nacional de México, al Instituto Tecnológico de Tijuana y a sus respectivas autoridades.

Cesar Alan Torres Quiroz

Cesar Torres.

Nombre completo del estudiante y firma autografa
Estudiante de la Maestría en Ciencias de la Computación

Resumen

Los modelos de redes neuronales convolucionales han demostrado capacidades increíbles al realizar tareas de clasificación, a pesar de esto, se han encontrado múltiples adversidades a la hora de aplicarlos en ambientes de dominio o áreas específicas, tales como lo son los sistemas de clasificación de señales de tránsito. La utilización de técnicas difusas para preprocesamiento de información muestra la capacidad de manejar incertidumbre encontrada en los datos que pasan por la red. En el presente trabajo de tesis se presenta la iniciativa, para realizar una incorporación de técnicas de lógica difusa dentro de un modelo de red neuronal convolucional para manejar la incertidumbre presente en las fuentes de información que pasan por el modelo a la hora de entrenarlo. En la implementación se plantea la utilización de información y filtros en el espectro difuso, así como la creación de una nueva capa para sustituir la capa clásica de convolución por un nuevo filtrado de ventana en el cual se aplican procesos de composición matricial. El trabajo fue probado en bases de datos orientadas a clasificación de señales de tránsito, debido a su complejidad, dadas en las distintas circunstancias y factores alternos en los que se puede encontrar un señalamiento.

Abstract

Convolutional neural network models have shown incredible capabilities when performing classification tasks, despite this, multiple adversities have been found when applying them in domain specific areas or environments, such as traffic sign classification. The utilization of fuzzy techniques for data preprocessing, shown capabilities of managing uncertainty found in the data sources that the network manages. In this thesis, we present the initiative, to incorporate fuzzy logic techniques inside a convolutional neural network to manage uncertainty present in the multiple data sources that the model handles when training. The implementation considers the use of information and filters in the fuzzy spectrum, as well as the creation of a new layer to replace the traditional convolution layer for a new window filtering layer where it applies matrix composition. The work was tested in multiple databases oriented to traffic signs, due to its complexity of the different circumstances and factors in where a traffic sign can be found.

Dedicatoria

A mi madre Estela, mi padre Cesar y mi hermana Danixia, por su amor y apoyo constante durante todo este proceso.

A mis queridos abuelos Aurora, Ambrocio y Francisca, que han sido como segundos padres para mí y siempre han estado ahí.

A mi primo y mentor Jesse Rodríguez, por su guía y apoyo incondicional en mi camino hacia mis metas.

Y a mis amigos, por su amistad y por estar ahí para escucharme y apoyarme en momentos difíciles.

A todos ustedes, gracias por hacer este viaje tan especial.

Agradecimientos

Al Instituto Tecnológico de Tijuana por brindar las instalaciones y los medios para realizar estudios de posgrado de calidad.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindar el apoyo económico con la beca número 1106407 para realizar mis estudios de posgrado.

A mi directora de tesis Claudia por permitirme la libertad de elegir un tema de tesis y por todos sus aportes a este trabajo de tesis y a mi formación.

A mis compañeros de clase y laboratorio por hacer esta trayectoria más divertida.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1. Introducción

Los avances en el área de cómputo inteligente han permitido la automatización relacionada a la localización y clasificación de objetos en entornos digitales, las cuales han mostrado ser útiles en diversas áreas. Las redes neuronales convolucionales son una rama del aprendizaje profundo, la cual se encuentra inspirada en la visión humana, lo que las hace competentes para la resolución de este tipo problemas debido a la capacidad de extracción de características y reconocimiento de patrones en imágenes.

Dada la necesidad de obtener un mejor rendimiento a la hora de utilizar sistemas inteligentes, se ha experimentado con la hibridación de diversas áreas de cómputo inteligente para mejorar estos resultados. Procesos como el preprocesamiento de imágenes o la utilización de sistemas difusos nos permiten mejorar este tipo de sistemas.

La lógica difusa es un área del cómputo inteligente, en la cual se busca efectuar una representación no lineal similar a la que realizamos los humanos, tomando en cuenta niveles de incertidumbre, la cual puede presentarse en las distintas fuentes de información dadas las diferentes escenas o ambientes en los cuales son capturados, así como los objetos de interés.

En la actualidad existe una gran cantidad de problemas los cuales son resueltos utilizando técnicas de aprendizaje profundo. Tareas como detectar enfermedades en radiografías [1], sistemas de identificación o seguimiento para un piloto automático en vehículos autónomos [2]. Esto ha llevado a que se trabaje en el estado del arte para crear múltiples modelos de aprendizaje profundo los cuales han sido entrenados para ser eficientes en múltiples áreas genéricas. Sin embargo, muchos problemas requieren conocimiento específico el cual hace que modelos existentes no den el rendimiento requerido para llevar a cabo las tareas de forma eficiente.

En los últimos años, los avances tecnológicos han logrado que las comunidades se adapten al uso de aplicaciones inteligentes, dado esto se ha visto la necesidad de incrementar la eficiencia de sistemas inteligentes en tareas específicas. Actualmente se ha optado por experimentar con hibridación de técnicas de cómputo suave y aprendizaje profundo para dar una mayor apertura a modelos más complejos para obtener mejores resultados. Estos modelos complejos a pesar de ser eficientes carecen de la capacidad de poder ser arbitrados en su proceso de toma de decisiones lo cual los hace poco aptos para algunas áreas donde es necesario tener una explicación de cómo es que se llega a un resultado. Debido a esto, se requiere investigar las distintas formas en que es

posible realizar la hibridación de técnicas, no solo para mejorar la eficiencia, si no lograr darles la capacidad a los modelos de generar resultados explicables.

Las cualidades de los sistemas de aprendizaje profundo pueden llegar a considerar millones de parámetros para realizar la toma de decisiones, estos sistemas son usualmente considerados como modelos de caja negra donde se desconoce la forma en que estos operan. Por lo cual el área de la inteligencia artificial explicable (XAI) busca dar un entendimiento a como estos modelos de caja negra toman decisiones, dando una justificación de los factores y el impacto que tuvieron estos a la hora de deliberar un resultado. La combinación de técnicas, tales como las redes neuronales convolucionales y lógica difusa nos permite ampliar el horizonte a la hora de obtener resultados competentes y explicables al realizar tareas de localización y/o clasificación, a diferencia de modelos pre-entrenados los cuales deliberan resultados de ser expuestos a enormes cantidades de información genérica.

El objetivo de esta investigación es desarrollar un nuevo modelo de aprendizaje profundo el cual combine los principios de redes neuronales convolucionales y la teoría de lógica difusa para resolver problemas de clasificación aplicado a casos de estudio de señales de tránsito. Concretamente esta meta se realizará concluyendo los siguientes puntos:

- Analizando los modelos de inteligencia artificial que combinan técnicas de redes neuronales profundas y lógica difusa.
- Aplicando técnicas de preprocesamiento de imágenes (tanto clásicas como difusas) a las distintas bases de datos utilizadas a lo largo del proyecto.
- Desarrollar un modelo de aprendizaje profundo el cual incorpore dentro de la arquitectura de una red neuronal convolucional estrategias de lógica difusa.
- Aplicar el modelo propuesto a problemas de prueba (benchmark) de clasificación de señales de tránsito.
- Realizar estudio comparativo y análisis estadístico del modelo propuesto, contra otros existentes.

El siguiente trabajo de tesis consta de siete capítulos, los cuales se describen a continuación.

capítulo 2 antecedentes: contiene algunos de los trabajos similares, en los cuales se trabajó con técnicas de hibridación con redes neuronales y procesamiento aplicado en el estado del arte.

capítulo 3 marco teórico: En esta sección se discute la terminología básica utilizada para el desarrollo de tesis, así como se da una explicación de las técnicas utilizadas y su definición

matemática; en este caso se discuten temas tales como, técnicas de preprocesamiento, redes neuronales, redes neuronales convolucionales, lógica difusa e inteligencia artificial explicable.

capítulo 4 propuesta. En este capítulo se plantea la idea general sobre la innovación realizada en este trabajo de tesis, así como se detalla su funcionamiento e implementación.

capítulo 5 experimentación y resultados. Contiene los resultados obtenidos de las experimentaciones realizadas con las distintas propuestas planteadas en este presente trabajo.

capítulo 6 discusión de resultados. En esta sección se analizan a detalle los resultados obtenidos, para formar conclusiones coherentes sobre el trabajo desarrollado.

capítulo 7 conclusiones y trabajo futuro. Consta de las conclusiones obtenidas con el trabajo de tesis, así como implementaciones o trabajos futuros para complementar este trabajo.

Capítulo 2. Antecedentes

En este capítulo se describen algunos de los trabajos que influenciaron y sirvieron de referencia para hacer esta investigación posible. Entre ellas se incluyen trabajos de preprocesamiento de imágenes, los cuales incluyen técnicas de preprocesamiento difuso, así como sistemas híbridos que combinan modelos de aprendizaje profundo en conjunto con técnicas de lógica difusa.

Uno de los trabajos más influenciables en el área de clasificación de imágenes utilizando redes neuronales convolucionales es el presentado por Krizhevsky et al [3], los autores describen el desarrollo del modelo AlexNet el cual fue entrenado con la base de datos ImageNet [4] la cual contiene más de 1.2 millones de imágenes en 1000 clases distintas. El modelo propuesto por los autores contiene más de 60 millones de parámetros entrenables, cuenta con 5 capas de convolución y 3 capas densamente conectas. Fue validado con las 50,000 imágenes de validación consiguiendo un error del top-1 con 33.5% superando a los modelos del estado del arte significativamente. Este trabajo es pionero en el uso de procesadores gráficos (GPU's) para incrementar la velocidad de entrenamiento de los modelos.

El modelo ResNet presentado por He et al [5], es un modelo con arquitectura profunda constituida en bloques la cual cuenta con hasta 152 capas, sin sufrir degradación, un problema muy común en modelos de esa escala. El modelo de ResNet introduce la idea de arquitecturas residuales, las cuales permiten pasar información de una o más capas de la red. La arquitectura fue entrenada con la base de datos ImageNet consiguiendo un error del top-5 del 3.57%, superando a múltiples modelos siendo considerado como estado del arte en tareas de clasificación. La arquitectura ResNet se ha utilizado en múltiples tareas de reconocimiento de imágenes y se ha convertido en una opción popular como base para entrenar CNN profundas.

En el trabajo presentado por Korshunova [6] proponen la extensión de una red neuronal convolucional clásica, utilizando un sistema de agrupación difuso cuyo objetivo es permitir la superposición de clases. El sistema difuso propuesto consiste en una capa la que se encuentra después de la extracción de características (capa de convolución y agrupación) generando grupos los cuales pasan a las capas densamente conectadas para obtener las clasificaciones, mejorando significativamente los resultados de clasificación. Las pruebas se realizaron entrenando con la base de datos "Dogs vs Cats" y realizando afinación de parámetros al modelo vgg16 con la base de datos ImageNet aumentando la precisión significativamente.

Hsu et al. [7], proponen la combinación de una red neuronal convolucional con una red neuronal difusa para realizar la sustitución con las capas densamente conectadas que se encuentran después de las capas de convolución. Esto con el fin de lograr entrenar modelos con resultados eficientes al utilizar bases de datos pequeñas, resolviendo la incertidumbre causada al tener pocos datos de entrenamiento. La capa difusa multicapa sustituye a las capas densamente conectadas utilizando funciones de membresía para el conjunto formado por cada clase. Las pruebas realizadas se hicieron con dos segmentos de la base de datos ImageNet, el primer segmento se creó utilizando tres clases de animales marinos. El segundo segmento utilizando tres clases de medios de transporte. Obteniendo resultados tangibles en las bases de datos de prueba de la experimentación realizada, aumentando la precisión hasta en un 7%.

La arquitectura YOLO presentada por Redmon et al [8], presentan una arquitectura única de red neuronal convolucional para predecir cajas delimitadoras y probabilidad de clase. El sistema se destaca en únicamente utilizar un modelo único para realizar la tarea, reduciendo el costo computacional y tiempo de predicción. YOLO fue entrenado utilizando las bases de datos PASCAL VOC [9] y COCO, demostrando que es capaz de superar a sistemas de detección en tiempo real en ambas bases de datos. El trabajo se destaca debido a su propuesta de cajas ancla, las cuales son un conjunto de delimitadores definidos en múltiples escalas y aspectos. Las anclas permiten que YOLO realice detección de objetos en múltiples tamaños y formas, reduciendo la cantidad de delimitadores predichos.

En el caso de Diamantis et al. [10], atacan la combinación de lógica difusa con redes neuronales convolucionales de una forma particular, generando una nueva capa de agrupación utilizando lógica difusa tipo 1. La capa propuesta por el autor busca combatir la incertidumbre presente en la información que pasa al modelo, tales como ruido, color, o ambigüedad geométrica. La capa opera sobre el conjunto de datos creado por el mapa de características generado después de la capa de convolución. La capa propuesta fue evaluada utilizando distintas bases de datos públicas. Se realizó una comparación con técnicas de agrupación comunes, tales como agrupación máxima, promedio, Regp, entre otras; consiguiendo un mayor porcentaje de precisión a la hora de clasificar en los conjuntos de las bases de datos utilizadas.

El trabajo realizado por Nguyen et al. [11], atacan de forma particular la hibridación de redes neuronales con lógica difusa. El autor propone una capa de convolución difusa unidimensional para extracción de características en texto y realizar análisis de sentimiento. Se realiza la

fusificación de las fichas obtenidas con el texto. En este caso aplica funciones de membresía a los tensores generados para aplicar la convolución con filtros e información difusa, generando mapas de características en el espectro difuso. La utilización de la convolución difusa propuesta se probó con cinco distintas bases de datos de procesamiento de lenguaje natural. Obteniendo mejores resultados comparados contra modelos que utilizan convolución clásica.

Capítulo 3. Marco Teórico

El siguiente capítulo se enfoca en presentar los conceptos teóricos esenciales para llevar a cabo la investigación en cuestión. En la Sección 3.1 se describen los aspectos relacionados con el preprocesamiento de imágenes digitales y los sistemas de detección de bordes. La Sección 3.2 se dedica a definir la arquitectura y componentes de las redes neuronales artificiales, incluyendo la estructura de las neuronas, las interconexiones, los métodos de entrenamiento y otros aspectos relevantes. La Sección 3.3 se enfoca en el concepto de redes neuronales convolucionales, incluyendo sus características y componentes. La Sección 3.4 se dedica a explicar los conceptos fundamentales de la lógica difusa. Finalmente, la Sección 3.5 presenta una descripción detallada del concepto de inteligencia artificial explicable.

3.1. Preprocesamiento de imagen

Para entrar en contexto con las técnicas de preprocesamiento de imágenes, es necesario definir el concepto de imagen. Las imágenes son una captura o mapeo bidimensional de un plano tridimensional. Esta es representada por la función $f(x, y)$ donde x y y son coordenadas espaciales con amplitud f. Las imágenes digitales manejan una representación de $f(x, y)$ finita y discreta [12]. La escena tridimensional es típicamente representada en una matriz de dos dimensiones de tamaño $M\ x\ N$ con M renglones y N columnas. La Ecuación (1) contiene una representación típica de una imagen digital.

$$f(m,n) = \begin{bmatrix} x(0,0) & \cdots & x(0,N-1) \\ \vdots & \ddots & \vdots \\ x(M-1,0) & \cdots & x(M-1,N-1) \end{bmatrix} \tag{1}$$

Los elementos (píxeles) localizados en las coordenadas (m, n) contienen una representación proporcional al brillo de la escena al momento de la captura. Los valores se encuentran discretizados 0 hasta 255. En caso de una foto a color o RGB se encuentra un arreglo que contiene la siguiente estructura $f(m, n, o) = (255,255,255)$ donde cada valor de este arreglo representa respectivamente a R, G y B.

El preprocesamiento de imágenes digitales nos permite realizar manipulaciones con el fin de lograr múltiples objetivos, tal como resaltar o esconder patrones, realizar ajustes de iluminación o eliminar artefactos creados por agentes externos al dispositivo de captura [13]. La metodología

usualmente consiste en aplicar una operación con una ventana de píxeles (kernel), la cual pasa por la imagen, cambiando su contenido uniformemente creando una nueva imagen [14]. El proceso de convolución es descrito por la Ecuación (2) donde k representa el kernel y r, c representan las coordenadas para renglones y columnas y la variable f la imagen de entrada.

$$(f * k)\,(x, y) = \iint k(r, c) f(x - r, y - c)\; dr\; dc \tag{2}$$

La salida de la convolución es determinada tomando en cuenta los siguientes factores:

- Tamaño de la imagen de entrada: $ancho \times altura$ en pixeles.
- Padding: pixeles que se añaden al borde de la imagen antes de aplicar el filtro.
- Tamaño del filtro: $ancho \times altura$ en pixeles.
- Paso: número de pixeles que se desplaza la ventana durante el proceso de convolución.

El tamaño de la salida de una convolución de dos dimensiones se calcula utilizando la Ecuación (3).

$$Salida = \frac{(Entrada + 2 * Padding - filtro)}{Paso + 1} \tag{3}$$

3.1.1. Detección de bordes

Los sistemas de detección de bordes consisten en filtros pasa alta que resalta los detalles más finos de las imágenes. Los bordes son considerados zonas de alto contraste, donde usualmente contienen objetos que se encuentran presentes en las imágenes. La detección de bordes permite reducir una imagen a su mínima expresión, permitiendo reducir la cantidad de información necesaria la cual es procesada en un sistema. Los filtros son basados en la primera y segunda derivada de la imagen digital. Algunos populares detectores de bordes son:

- Detector de bordes de Canny [15].
- Detector de bordes Laplaciano [16] [17].
- Detector de bordes de Sobel [18].
- Detector de bordes de Prewitt. [19].

- Detector de bordes por gradiente morfológico [20].

En concreto, esta investigación se enfocó en sistemas de detección de bordes por Sobel, Prewitt y gradiente morfológico.

3.1.1.1. Detección de bordes por Sobel y Prewitt

La detección de bordes por método de Sobel y Prewitt son similares entre sí, dado a que estas consisten en operadores de gradiente con una ventana de tamaño 3×3 para los ejes x, y denominados como $Sobel_x$ Ecuación (4), $Sobel_y$ Ecuación (5) para el filtrado por Sobel y $Prewitt_x$ Ecuación (6), $Prewitt_y$ Ecuación (7) para detección por método de Prewitt. Estos operadores son aplicados a la imagen original mediante operaciones de convolución definida por la Ecuación (2).

$$Sobel_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4}$$

$$Sobel_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{5}$$

$$Prewitt_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{6}$$

$$Prewitt_{\mathrm{y}} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{7}$$

El resultado de la operación de convolución genera dos gradientes g_x Ecuación (8) y g_y Ecuación (9) los cuales representan el eje horizontal y vertical respectivamente. El valor del filtro se encuentra definido al gradiente, ya sea $Sobel_x$ o $Prewitt_x$ para g_x y $Sobel_y$ o $Prewitt_y$ para g_y La magnitud de los gradientes es denotada por G el cual es calculado con la Ecuación (10), este contiene la agregación ponderada de los filtros para ambos ejes.

$$g_x = \sum_{i=1}^{i=3} \sum_{j=1}^{j=3} filtro_{x_{i,j}} * f_{x+i-2,y+j-2} \tag{8}$$

$$g_y = \sum_{i=1}^{i=3}\sum_{j=1}^{j=3} filtro_{y_{i,j}} * f_{x+i-2,y+j-2} \quad (9)$$

$$G[f(x,y)] = \sqrt{g_x^2 + g_y^2} \quad (10)$$

3.1.1.2. Detección de bordes por gradiente morfológico

Al utilizar gradiente morfológico para detección de bordes se calcula la primera derivada de la imagen en las cuatro orientaciones de esta (horizontal 0°, vertical 90°, y ambas diagonales $45°, 135°$). Estos gradientes son denominados por las variables G_1, G_2, G_3 y G_4 mostrados por la Figura 3.1 (a).

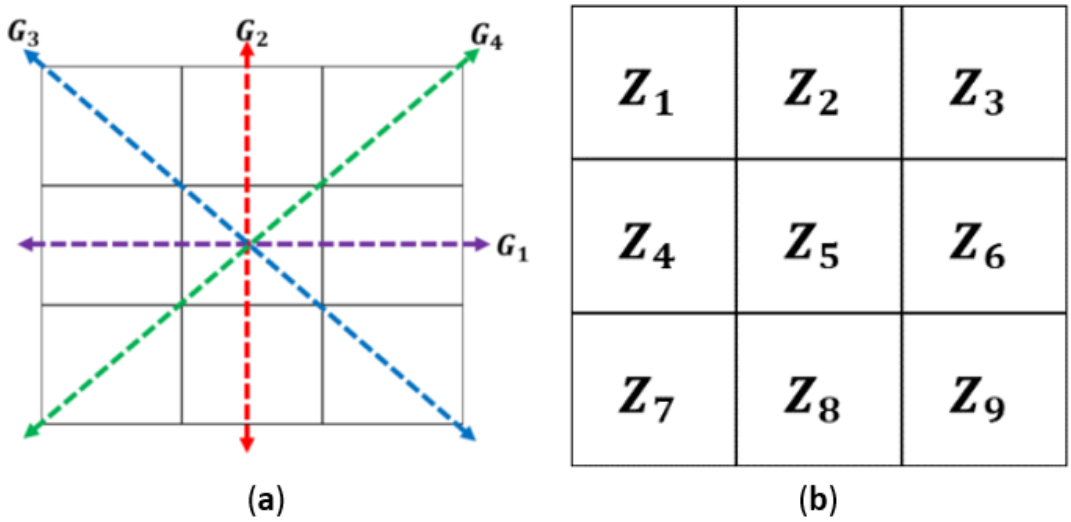

Figura. 3.1. (a) Dirección de gradientes, (b) Coeficiente Z_i.

La Ecuación (11) denota cómo se calculan los bordes en los gradientes G_1 a G_4 de la imagen,, mediante la utilización de una ventana de 3×3, la variables z_1 a z_9 representa los coeficientes de las posiciones denotadas por la Figura 3.1 (b), los cuales son calculados mediante la Ecuación (12) para finalmente ser sumados y resultar en los bordes de la imagen denotados por la Ecuación (13) [21] [22] [23].

$$
\begin{aligned}
G_1 &= \sqrt{(z_5 - z_2)^2 + (z_5 - z_8)^2}, \\
G_2 &= \sqrt{(z_5 - z_4)^2 + (z_5 - z_6)^2}, \\
G_3 &= \sqrt{(z_5 - z_1)^2 + (z_5 - z_9)^2}, \\
G_4 &= \sqrt{(z_5 - z_3)^2 + (z_5 - z_7)^2}
\end{aligned} \tag{11}
$$

$$
\begin{aligned}
z_1 &= f(x-1, y-1), \\
z_2 &= f(x, y-1), \\
z_3 &= f(x+1, y-1), \\
z_4 &= f(x-1, y), \\
z_5 &= f(x, y), \\
z_6 &= f(x+1, y), \\
z_7 &= f(x-1, y+1), \\
z_8 &= f(x, y+1), \\
z_9 &= f(x+1, y+1)
\end{aligned} \tag{12}
$$

$$
Edges = G_1 + G_2 + G_3 + G_4 \tag{13}
$$

3.2. Redes neuronales artificial

Las redes neuronales artificiales son una técnica del aprendizaje máquina en la cual se busca imitar la manera en que organismos biológicos aprenden. Basándose en las unidades de cómputo (neuronas), que se encuentran interconectadas entre sí para comunicarse (sinapsis) y responder a estímulos externos [24]. La intensidad de la conexión cambia dependiendo al tipo de estímulo al cual se sea sometido. Este cambio de conexiones conforma el proceso de aprendizaje en los organismos biológicos.

3.2.1. Neurona artificial

El término de neurona artificial fue desarrollado por Warren y Pitts en la década de los 40's [25]. Los inicios de la neurona artificial fueron planteados matemáticamente, la cual podría usarse para calcular cualquier función lógica. Las neuronas se presentaron como un dispositivo matemático capaz de realizar cálculos lógicos mediante el uso de una función de umbral binario.

Esta idea fue llevada a una escala mayor, demostrando que una red de neuronas artificiales podría ser utilizada para implementar cualquier función lógica, independientemente de su complejidad, mediante la representación matemática de su actividad. El trabajo propuesto por los autores discute la conexión de distintas neuronas, argumentando que estas podrían tener la capacidad de realizar el cómputo de operaciones más complejas. A este modelo matemático de la red de conexiones se le describe como feedforward, donde la información fluye en una sola dirección, desde los nodos de entrada hasta los nodos de salida, sin retroalimentación.

3.2.2. Perceptrón

El concepto de red neuronal es derivado del trabajo presentado por Frank Rosenblatt en su documento "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," a finales de la década de los 50's [26]. En su trabajo, Rosenblatt argumenta que los perceptrones pueden ser utilizados para modelar neuronas en el cerebro, replicando la habilidad de aprender y reconocer patrones, dado a esto el perceptrón fue modelado de un modelo neurológico biológico para realizar una tarea de clasificación. El modelo consiste en una arquitectura de doble capa (entrada y salida) con pesos y sesgos ajustables. El perceptrón recibe la entrada de un conjunto de neuronas y produce su salida la cual es calculada utilizando los pesos y sesgos. La Figura 3.2 contiene una comparativa entre una neurona biológica y una neurona artificial.

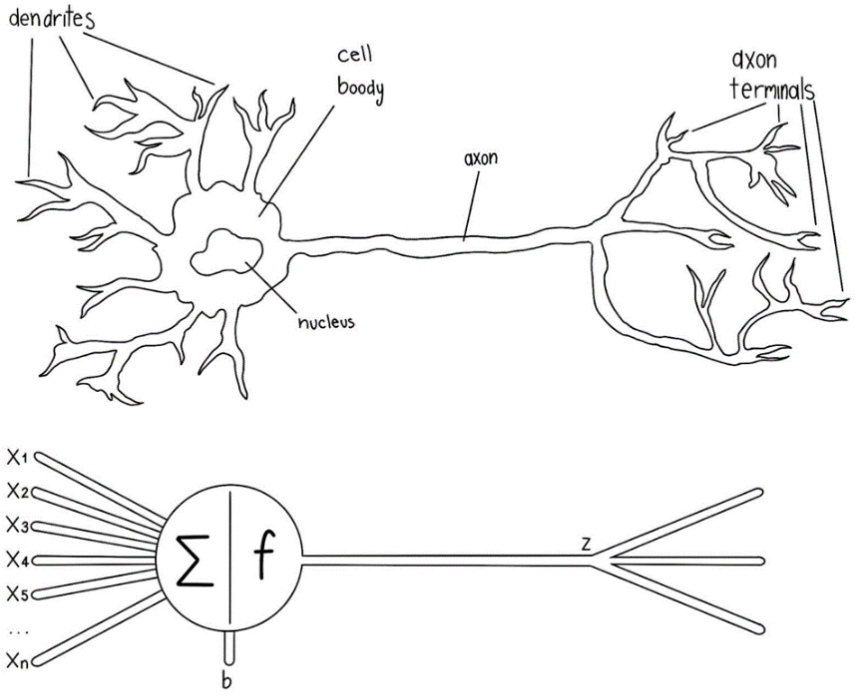

Figura. 3.2. Comparativa entre neurona biológica y neurona artificial.

El entrenamiento del perceptrón es realizado utilizando un algoritmo de aprendizaje supervisado llamado regla de aprendizaje del perceptrón. La regla ajusta los pesos y sesgos en función del error, el cual es calculado entre la salida obtenida y la salida real. El objetivo de este entrenamiento es encontrar un conjunto de pesos y sesgos los cuales logren clasificar de forma correcta la información de entrada.

El perceptrón calcula su salida realizando un producto punto de las entradas con los pesos, sumando el sesgo y luego pasando el resultado a través de una función de activación. El producto escalar de las entradas y los pesos se calcula multiplicando cada valor de entrada por su peso correspondiente y sumando todos los productos. Esto se puede representar matemáticamente como $\sum_{i=1}^{n} pesos_i \times entradas_i$, después de este paso se le agrega el sesgo, el cual actúa como peso adicional y es representado como $\sum_{i=1}^{n} pesos_i \times entradas_i + sesgo$. Una vez obtenido el producto de los pesos y el sesgo se utiliza la función de activación, comúnmente en el perceptrón se utiliza la función de paso. La función de paso toma una sola entrada de valor real y produce

una salida de 1 si la entrada es mayor que cierto umbral (generalmente 0) y una salida de -1 si la entrada es menor que el umbral. Dado esto, el cálculo de la salida del perceptrón puede ser resumido con la Ecuación (14).

$$salida = activación\left(\sum_{i=1}^{n} pesos_i \times entradas_i + sesgo\right) \tag{14}$$

3.2.3. Aprendizaje profundo

El área de aprendizaje profundo es una rama del cómputo suave, el cual se enfoca en la creación de modelos de redes neuronales artificiales con múltiples capas de procesamiento, conocidas como redes neuronales profundas (Deep Neural Networks, DNNs), para modelar patrones y relaciones complejos en los datos. Estas redes son capaces de aprender representaciones automáticas a través de las capas, permitiendo a los modelos aprender directamente de los datos, sin necesidad de una ingeniería de características manual.

El concepto de aprendizaje profundo fue inspirado en el cerebro humano, en la forma en que este procesa información, donde las capas de neuronas procesan y transmiten la información a través de la red, realizando así la identificación de patrones dados los estímulos a los que es expuesta.

Las técnicas de aprendizaje profundo han tenido un gran impacto en campos como la visión por computadora, el procesamiento del lenguaje natural y el aprendizaje automático en general, ya que permite a las computadoras aprender de forma autónoma a partir de datos sin la necesidad de programación específica. Los usos de esta tecnología incluyen, sistemas de reconocimiento de voz y rostros, traducción automática, sistemas de navegación autónoma, entre otros.

El término de aprendizaje profundo fue definido por Bengio, LeCun y Hinton [27]. La definición dada por los autores describe que la utilización de redes feedforward con una estructura simple de capa única, tiene limitantes al modelar patrones complejos; según los autores, las redes neuronales profundas son capaces de aprender representaciones de los datos a niveles cada vez más abstractos a medida que se avanza a través de las capas intermedias o "escondidas" de la red. Esto permite a la red aprender patrones cada vez más complejos y abstractos a medida que se analizan los datos en múltiples niveles de abstracción.

La utilización de capas intermedias permite aprender de representaciones no lineales de los datos, dándoles las capacidades adecuadas para la resolución de problemas complejos. De igual

forma, los modelos de aprendizaje profundo son capaces de aprender de forma automática características relevantes de los datos, reduciendo la necesidad de selección manual de características.

Un sistema de aprendizaje profundo generalmente está compuesto de los siguientes componentes principales:

- Capas de la red: Una red neuronal profunda está compuesta por varias capas de nodos o "neuronas", cada una de las cuales está conectada a las capas anteriores y posteriores. Las capas internas se denominan capas ocultas, y las capas externas se denominan capas de entrada y salida.
- Funciones de activación: En cada capa de la red se utilizan funciones de activación para controlar el flujo de información entre las neuronas. Algunas funciones comunes son sigmoidales, ReLU, tangencial, entre otras [28]. En la Figura 3.3 se observan las gráficas de estas funciones, las Ecuaciones (15-20) contienen la función utilizada para desarrollar las gráficas.
- Método de retro propagación: Es un algoritmo de optimización que se utiliza para ajustar los pesos de las conexiones entre las neuronas de forma automática. Es el método más común para entrenar una red neuronal profunda. Algoritmos de entrenamiento hacen uso de la retro propagación para calcular los gradientes con respecto al error en los pesos de la red, agregando técnicas de optimización para hacer el proceso aún más eficiente, Algoritmos comunes de entrenamiento incluyen gradiente estocástico descendiente (SDG) [29] o Adam [30].
- Funciones de pérdida o costo: Es una función matemática que se utiliza para medir el error entre la salida esperada y la salida obtenida por la red. El objetivo del entrenamiento es minimizar esta función de pérdida [31], a continuación, se describen algunas de las funciones típicamente utilizadas.
    - Error Cuadrático Medio (MSE) para problemas de regresión, que es la media del cuadrado de la diferencia entre el valor obtenido y el valor real.
    - Entropía cruzada categórica para problemas de clasificación multiclase, que mide la disimilitud entre la distribución de probabilidad obtenida y la distribución verdadera.

- Entropía cruzada binaria para problemas de clasificación binaria, es similar a la entropía cruzada categórica pero solo para dos clases.

- Conjunto de datos de entrenamiento y prueba: Es necesario tener un conjunto de datos para entrenar la red y evaluar su desempeño. La calidad y cantidad de datos son fundamentales para mejorar el desempeño de la red.

Además, hay algunos otros componentes o técnicas que pueden utilizarse para mejorar el rendimiento de una red neuronal profunda, como regularización, capa de desprendimiento, normalización de lote, entre otros.

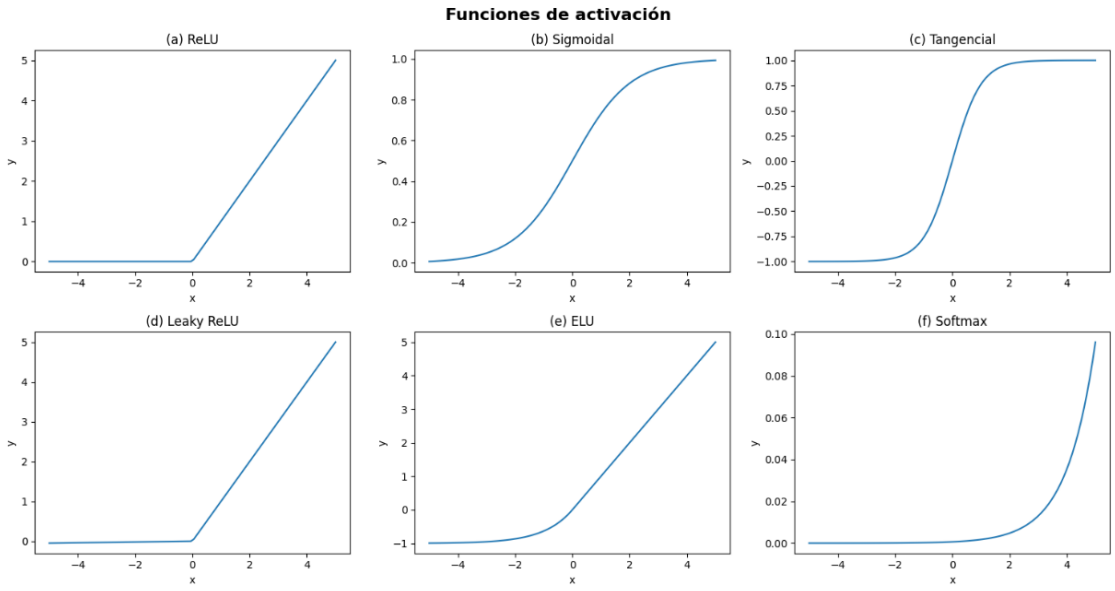

Figura. 3.3. Funciones de activación (a) ReLU, (b) Sigmoidal, (c) Tangencial, (d) Leaky ReLu, (e) ELU, (f) Softmax.

$$ReLU(x) = max\begin{cases}0\\x\end{cases} \tag{15}$$

$$Sigmoidal(x) = \frac{1}{1+e^{-x}} \tag{16}$$

$$Sigmoidal(x) = \frac{1}{1+e^{-x}} \tag{17}$$

$$Leaky\ ReLU(x) = \begin{cases}0.01x, & \text{para } x < 0\\ x, & \text{para } x \geq 0\end{cases} \tag{18}$$

$$ELU(x) = \begin{cases}x, & \text{para } x \geq 0\\ \alpha(e^x - 1), & \text{para } x < 0\end{cases} \tag{19}$$

$$Softmax(x_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \text{para } i = 1, \dots, K \tag{20}$$

En la Figura 3.4 podemos observar un diagrama típico de una red neuronal artificial.

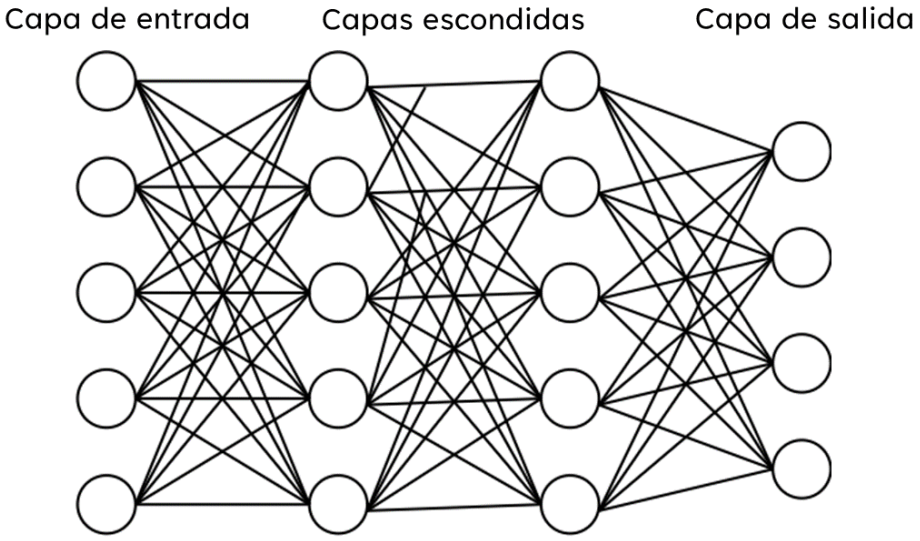

Figura. 3.4. Arquitectura básica de una red neuronal artificial.

3.3. Redes neuronales convolucionales

Una red neuronal convolucional (CNN) es un método popular del aprendizaje profundo en el cual el sistema imita la corteza visual en conjunto a los estímulos que esta genera en el cerebro para generar patrones visuales [32]. La técnica utilizada es la "convolución" en la que se aplican ventanas (kernels) aleatorios para realizar extracción de patrones de la información. Estas redes son comúnmente diseñadas para procesar datos con una estructura espacial, usualmente utilizadas en tareas de visión computacional, como la clasificación y el procesamiento de imágenes. El término convolucional se deriva de la utilización de la fórmula de convolución Ecuación (21), donde:

- f: Imagen de entrada.
- k: kernel (filtro) de entrada.
- i: columna de la imagen.
- j: renglón de la imagen.

- r: renglón del filtro.
- c: columna del filtro.
- $(f * k)$: imagen resultante.

$$(f * k)[i,j] = \sum_{r=0}^{n} \sum_{c=0}^{m} k[r,c] \times f[i-r, j-c] \tag{21}$$

La Figura 3.5 muestra una representación gráfica de una red neuronal convolucional con sus principales capas.

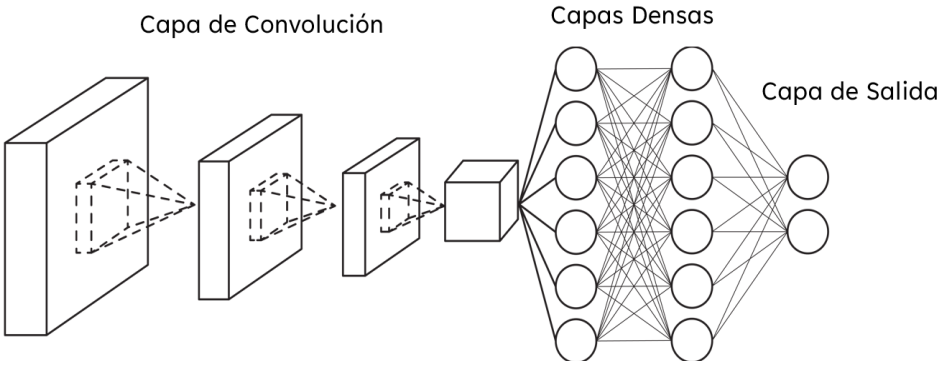

Figura. 3.5. Arquitectura básica de una red neuronal convolucional.

Las CNNs consisten en un modelo de capas similar a la red neuronal artificial, estos modelos constan de varias capas de convolución, seguidas de capas de "agrupamiento" (también conocidas como "pooling"). La arquitectura básica de una CNN está formada de las siguientes capas: capa de entrada, capa convolucional, función de activación, función de agrupamiento, capa densamente conectada y capa de salida. A continuación, se describen a detalle cada una de las siguientes capas del modelo.

- Capa de entrada: La capa de entrada consiste en la receptora de información, usualmente consta de medidas ya definidas (estáticas).
- Capa de convolución: Esta es la capa fundamental en una CNN. Utiliza un conjunto de filtros aleatorios que se aplican a los datos de entrada para detectar patrones específicos en los datos. Cada filtro se desliza sobre los datos de entrada y produce una "activación" cuando detecta un patrón específico, generando una nueva matriz la cual contiene los contenidos extraídos. Estas activaciones se utilizan como entrada para la siguiente capa de la red. La Figura 3.6 contiene una

ejemplificación del proceso de convolución aplicando un filtro aleatorio a una matriz.

Similarmente a los pesos clásicos de las capas densamente conectadas, los filtros aleatorios formados en la capa de convolución son entrañables. Durante el entrenamiento del modelo, los pesos de estas matrices son ajustados para disminuir el error entre la salida verdadera y esperada. Los ajustes de estos pesos son determinados por el algoritmo de retro propagación y la función de optimización elegida en el modelo.

- Función de activación: La función de activación determina si una neurona debería o no ser activada. La función de activación de Unidad Lineal Rectificada (ReLu) es una utilizada ampliamente. La función ayuda a capturar y aprender de relaciones complejas de la información de entrenamiento, actuando como función lineal en valores positivos y no lineal en los valores negativos.
- Capa de agrupación: Esta capa nos permite realizar una representación reducida de la información, reduciendo la cantidad de parámetros que esta contiene. La agrupación máxima (Max Pooling) representa un filtro de tamaño $n \times n$, el cual recorre la información similar al proceso de convolución. En este caso a diferencia de aplicar un producto escalar, se extrae el valor máximo de esta ventana para almacenarlo en una nueva de la representación, reduciendo las dimensiones proporcionalmente al tamaño del filtro que se utilice, así como el tamaño del paso similarmente a la capa de convolución, el tamaño de salida se calcula con la formula $Salida = (Entrada - filtro) / paso + 1$, en este caso omitiendo el paddign debido a que se busca disminuir el tamaño de la entrada. En la Figura 3.7, se puede observar una representación de la función de Max pooling aplicada a una matriz sencilla.
- Capa de expulsión: Es una técnica de regularización que consiste en desactivar al azar un porcentaje de las neuronas en cada capa durante el entrenamiento. Esto ayuda a prevenir el sobreajuste y mejora el rendimiento del modelo.
- Capa densamente conectada: Consiste en los pesos y sesgos similares a redes neuronales tradicionales. La entrada de la capa consiste en la salida de la capa de

convolución en forma de un vector de características de las capas anteriores y lo transforma en una salida específica, como una clasificación de la imagen.

- Capa de salida: Es la última capa de una red neuronal, en la que se realiza la clasificación o la regresión. Dependiendo del problema, esta capa puede tener una o varias salidas.

Para realizar una CNN es necesario realizar una composición de varias capas de convolución con funciones de activación y capas de agrupamiento, seguidas de una o varias capas completamente conectadas y una capa de salida. La elección y configuración de las capas dependerá del problema específico y de los datos de entrada.

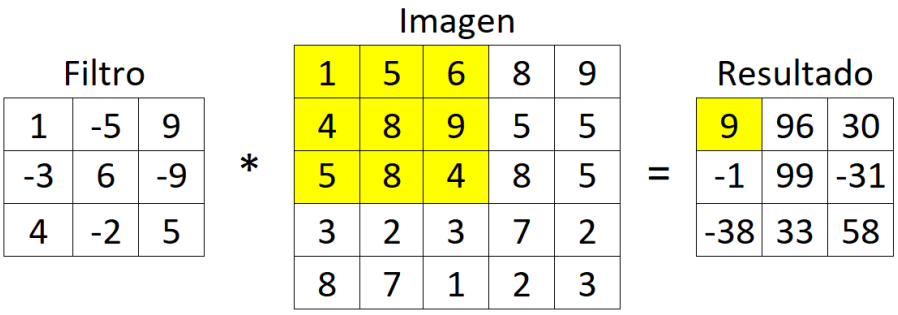

Figura. 3.6. Función de convolución en dos dimensiones.

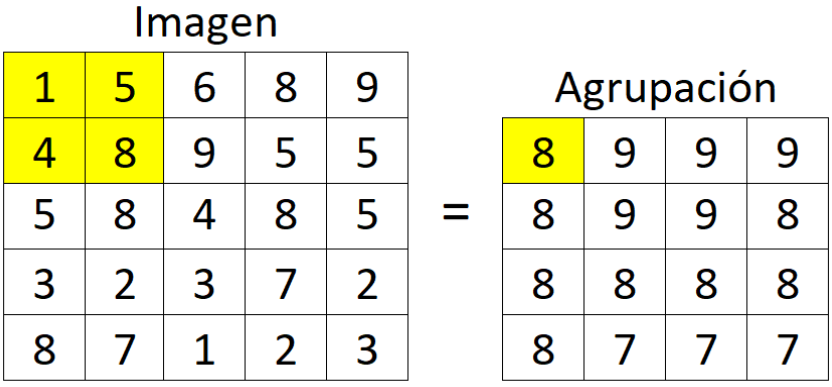

Figura. 3.7. Función de agrupación máxima (Max Pooling) utilizando filtro $2x2$ y $paso = 1$.

3.4. Lógica difusa

La lógica difusa, es un área del cómputo inteligente la cual tiene como objetivo utilizar modelos representativos no lineales, estas representaciones buscan tomar en cuenta niveles de incertidumbre que se encuentran en las distintas fuentes de información, similarmente al proceso de pensamiento humano donde existen conceptos imprecisos con grados de verdad.
La implementación de lógica difusa tiene aplicaciones en diversas disciplinas, tales como la robótica, control de procesos, medición de calidad y sistemas de toma de decisiones en sistemas expertos. En aplicaciones de control de procesos se utiliza para manejar sistemas complejos como los sistemas de control de climatización y los sistemas de control de motores. En inteligencia artificial se emplea para crear sistemas expertos capaces de razonar con imprecisión e incertidumbre, los sistemas expertos se extienden en variedad de campos, tales como la medicina, ingeniería y finanzas, entre muchos otros.

3.4.1. Conjuntos difusos

La lógica difusa se encuentra basada en los conceptos de los conjuntos difusos propuestos por Lotfi A. Zadeh [33] en la década de los 70s, el término fue acuñado debido a las carencias de los conjuntos tradicionales al realizar modelado de incertidumbre, que está presente en muchos casos cotidianos. Estos conjuntos tienen la particularidad en la que permiten realizar representaciones en un espacio continuo, a diferencia de los conjuntos clásicos (crisp sets) todos los elementos en el conjunto son incluidos o excluidos, estado que es irrealista para múltiples casos reales.
Los conjuntos difusos utilizan funciones de membresía (MF), las que permiten que un elemento sea parte de más de una clase simultáneamente. Los elementos de los conjuntos difusos mantienen grados de pertenencia en un intervalo de $[0,1]$, donde 0 representa total exclusión y 1 total inclusión. El rango de los elementos en un conjunto difuso está definido por las funciones de membresía donde sus rangos son definidos por expertos en el área. Las variables de membresía hacen referencia a conceptos lingüísticos que incluyen grados de incertidumbre. La Figura 3.8 contiene una representación de un conjunto difuso con la pertenencia y universo de discurso.

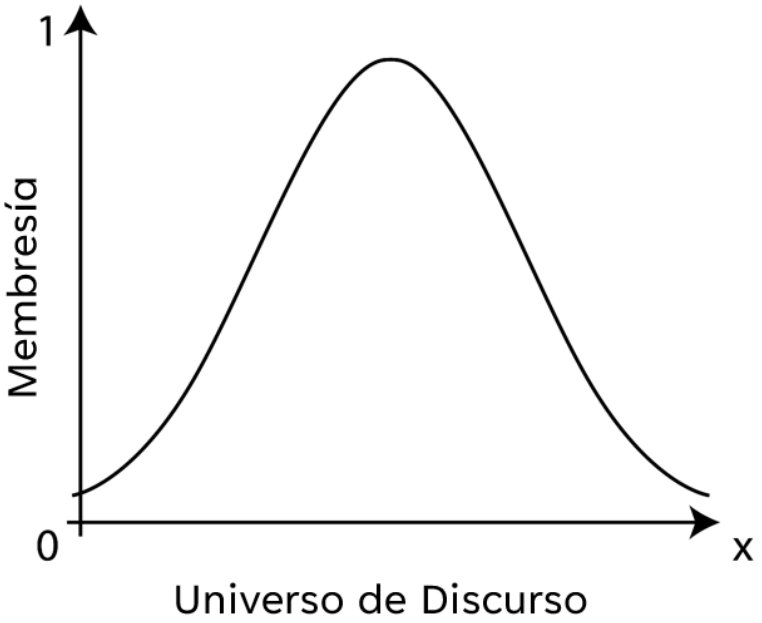

Figura. 3.8. Ejemplo de membresía en conjunto difuso.

Una ejemplificación de conjunto difuso de tipo 1 es representada por A, donde es un conjunto en el $Universo\ X$, típicamente definido como universo de discurso [34], tiene valores entre $[0,1]$, el cual pertenece a la función continua. $\mu_A: X \rightarrow [0,1]$. La función de membresía de tipo 1 de A es denotada como $\mu_A(x)$, el conjunto difuso denotado como pares ordenados es representado por la Ecuación (22).

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{22}$$

Para realizar el manejo de la incertidumbre, los sistemas difusos hacen implementación de términos lingüísticos para representar calores, tal como una expresión verbal, esto para describir un grado de pertenencia con los elementos del conjunto difuso. Los términos lingüísticos se relacionan con las funciones de membresía, que especifican el grado de pertenencia del universo. Estas funciones de pertenencia suelen ser curvas continuas y suaves, como una función gaussiana o trapezoidal, que describen cómo la pertenencia a un conjunto difuso cambia con el valor numérico de la variable.

Algunas de las funciones de membresía más comunes para representar conjuntos difusos son las siguientes: Triangular Ecuación (23), Trapezoidal Ecuación (24), Gaussiana Ecuación (25) o S Ecuación (26). La Figura 3.7 (A) – (D) contiene las gráficas de las Funciones de Membresía antes descritas.

$$Función\ de\ membresía\ Triangular:\ f(x;a,b,c)\begin{cases}0, & x \le a\\ \frac{x-a}{b-a}, & a \le x \le b\\ \frac{c-x}{c-b}, & b \le x \le c\\ 0, & c \le x\end{cases} \quad (23)$$

$$Función\ de\ membresía\ Trapezoidal:\ f(x;a,b,c,d) = \max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \quad (24)$$

$$Función\ de\ membresía\ Gaussiana:\ f(x,\sigma,c) = e^{\frac{-(x-c)2}{2\sigma^2}} \quad (25)$$

$$Función\ de\ membresía\ S:\ f(x;a,b)\begin{cases}0, & x \le a\\ 2\left(\frac{x-a}{b-a}\right), & a \le x \le \frac{a+b}{2}\\ 1-2\left(\frac{x-a}{b-a}\right)^2, & \frac{a+b}{2} \le x \le b\\ 1, & x \ge b\end{cases} \quad (26)$$

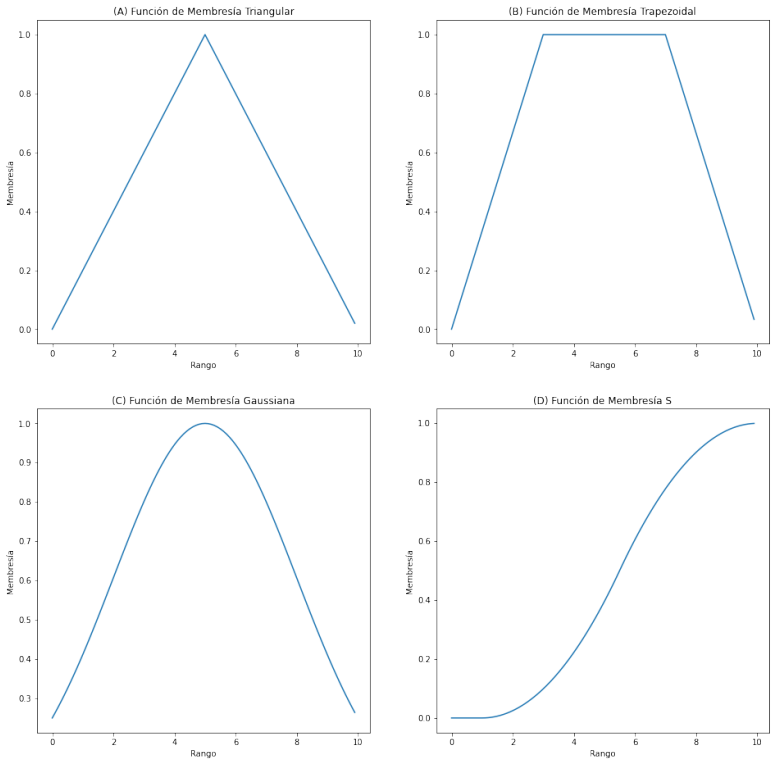

Figura. 3.9. (a) Función de membresía Triangular, (b) Función de membresía Trapezoidal, (c) Función de membresía Gaussiana, (d) Función de membresía S.

Similar a los conjuntos tradicionales, los conjuntos difusos permiten realizar operaciones básicas, en este caso se sustituyen los operadores por operadores difusos, tales como la unión, intersección, subconjuntos y complementos.

La unión de conjuntos denota una función cuya pertenencia sea de al menos uno de los conjuntos difusos con un grado de pertenencia dado. Sean A y B conjuntos difusos y C el resultado de la unión denotado como $C = A \cup B$ Tenemos que la unión se representa por la Ecuación (27).

$$\mu_C(x) = max\big(\mu_A(x)\big), \mu_B(x) = \mu_A(x) \vee \mu_B(x) \tag{27}$$

La intersección de los conjuntos difusos permite realizar la agregación de dos conjuntos A y B por la función $T: [0,1] \times [0,1] \rightarrow [0,1]$, la cual realiza la agregación de las funciones por la Ecuación (28). Donde $\wedge$ representa un operador binario para la función T. La intersección se utiliza para obtener el conjunto de elementos que pertenecen a ambos conjuntos difusos con un grado de pertenencia dado.

$$\mu_{A \cap B}(x) = T\big(\mu_A(x)\big), \mu_B(x) = \mu_A(x) \wedge \mu_B(x) \tag{28}$$

Un subconjunto difuso es un conjunto difuso que se encuentra dentro de otro conjunto difuso y que se define mediante un grado de pertenencia. Este se encuentra denotado por la Ecuación (29).

$$A \subseteq B \Leftrightarrow \ \mu_A \leq \ \mu_B \ para \ toda \ x \tag{29}$$

El operador de complemento es una función continua $N: [0,1] \rightarrow [0,1]$ la cual cumple con los siguientes requerimientos denotados por la Ecuación (30).

$$\begin{gathered} N(0) = 1 \text{ y } N(1) = 0 \\ N(a) \geq N(b) \ si \ a \leq b \end{gathered} \tag{30}$$

3.4.2. Relaciones difusas

Las relaciones difusas son conjuntos los cuales mapean elementos en sus conjuntos a un grado de pertenencia. Generalmente se utilizan para representar la incertidumbre o la imprecisión en las relaciones entre objetos o conceptos mediante grados de pertenencia, que son valores numéricos comprendidos entre 0 y 1 que indican la probabilidad o el nivel de certeza de que un objeto o un concepto está relacionado con otro. La representación de la relación binaria difusas se da por la

Ecuación (31) donde X, Y son universos de discurso donde $\mu_{\mathcal{R}}$ es bidimensional, dado esto es convencional realizar la representación de la relación $\mathcal{R}$ como una matriz de relación.

$$\mathcal{R} = \{((x, y), \mu_{\mathcal{R}}(x, y)) | (x, y) \in X \times Y\} \tag{31}$$

3.4.3. Composición de relaciones

Dadas las relaciones difusas, es posible realizar composiciones entre varias relaciones para obtener una relación difusa compuesta. La composición es una técnica clave en sistemas de inferencia difusa. Entre las técnicas de composición más populares se encuentra la $composición\ max - min$ y $composición\ max - product$.

- $Composición\ max - min$: Esta relación se basa en el uso de los operadores lógicos "and" y "or" difusos, que se corresponden con la intersección y la unión difusa, respectivamente. Dadas dos relaciones $\mathcal{R}_1$ y $\mathcal{R}_2$ por dos relaciones difusas definidas como $X \times Y$ y $Y \times Z$, respectivamente, la composición se encuentra definida por la Ecuación (32).

  $$\mathcal{R}_1 \circ \mathcal{R}_2 = \left\{ \left[(x, z), \max \min \left(\mu_{\mathcal{R}_1}(x, y), \mu_{\mathcal{R}_1}(y, z) \right) \right] | \ x \in X, y \in Y, z \in Z \right\} \tag{32}$$

- $Composición\ max - product$: El resultado de esta composición se obtiene utilizando el máximo del producto de los grados de pertenencia de las relaciones difusas involucradas para obtener el grado de pertenencia de la relación compuesta, la Ecuación (33) explica a detalle esta relación.

  $$\mu_{\mathcal{R}_1 \circ \mathcal{R}_2} = (x, z) = \max \left[\mu_{\mathcal{R}_1}(x, y), \mu_{\mathcal{R}_2}(y, z) \right] \tag{33}$$

1.1.1 Reglas If-Then

Las reglas difusas If-Then (Si-Entonces) son enunciados condicionales que describen la relación entre variables de entrada y salida en términos de grados donde existe un antecedente y una consecuencia. La Ecuación (34) contiene la forma general de las reglas difusas, donde A, B son variables lingüísticas y en los universos de discurso X, Y. El antecedente toma la forma de si x es A, mientras que la consecuencia toma la forma y es B. Esta relación comúnmente puede ser abreviada por la forma $A \rightarrow B$ denotando a la relación entre las variables.

$$\begin{gathered} \text{si } x \text{ es } A \text{ entonces } y \text{ es } B \\ R = A \rightarrow B = A \times B \end{gathered} \tag{34}$$

1.1.1 Razonamiento difuso

La regla básica de inferencia difusa se encuentra inspirada en *modus ponens*, llamado *modus ponens generalizado*, ya que toma en cuenta razonamiento aproximado. En este caso la inferencia se realiza utilizando múltiples premisas para obtener la consecuencia, esto se encuentra descrito por la Ecuación (35), Las variables A, B, A', B' son conjuntos difusos y donde A' es cercano a A y B' es cercano a B.

$$\begin{aligned} &premisa\ 1\ (hecho): && x = A', \\ &premisa\ 2\ (regla): && si\ x = A \rightarrow y = B, \\ &consecuencia\ (conclusión): && y = B' \end{aligned} \tag{35}$$

3.4.4. Sistemas de inferencia difusa

Los sistemas de inferencia difusa (FIS) son mecanismos los cuales toman ventaja de la lógica difusa para realizar la toma de decisiones o predicciones basados en la incertidumbre de la información. Los sistemas están basados en reglas If-Then con razonamiento difuso [34], utilizando antecedentes (If) y precedentes (Then) para esta toma de decisiones y llegar a un resultado. Los sistemas de inferencia difusa evalúan las reglas con los antecedentes y utilizan los consecuentes de las reglas relevantes para realizar el proceso de agregación y generar una salida pertinente. El flujo de datos en un sistema de inferencia puede ser visualizado en la Figura 3.10. Los componentes del sistema son descritos a continuación.

- Conjuntos difusos: Son los conjuntos de variables lingüísticas, los cuales describen los conceptos utilizados en el sistema.
- Base de datos: contiene el conocimiento (reglas) de expertos en el área, estas reglas determinan el comportamiento del sistema basándose en sus entradas y salidas.
- Funciones de membresía: Las funciones que describen el valor de la membresía dada a cada entrada al sistema, perteneciendo a un conjunto difuso. Los rangos de estas funciones son igual determinadas por expertos en el área, similar a las bases de datos.

- Mecanismo de razonamiento: Este componente del sistema se encarga de aplicar las reglas difusas a las entradas y generar una salida. Utiliza las funciones de pertenencia para determinar el grado de verdad de los antecedentes de las reglas y el grado de verdad de los consecuentes. Algunos de los mecanismos de inferencia difusa más populares son los sistemas de Mamdani [35], Tsukamoto [36], y Takagi-Sugeno-Kang [37].

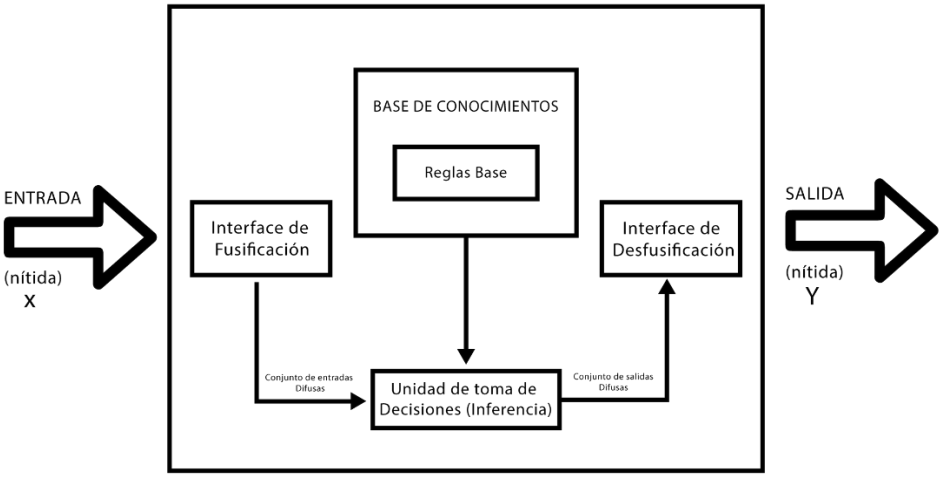

Figura. 3.10. Ejemplo de sistema de inferencia difusa.

3.5. Inteligencia artificial explicable

El término de inteligencia artificial explicable (XAI) surgió de la necesidad de un área o concepto en el cual se pudiera dar un entendimiento a los modelos de clasificación que funcionan con modelos de aprendizaje profundo, que suelen ser considerados como modelos de caja negra, esto debido a la complejidad de su funcionamiento al realizar tareas de inferencia [38]. El área de la XAI busca dar el acercamiento humano necesario para que modelos complejos logren ser auditables, dando explicaciones a factores que se tomen en la información para realizar toma de decisiones, dando un entendimiento sobre fortalezas y debilidades que pueda tener el modelo, así ofreciendo una oportunidad de mejorarlo para que pueda ser aplicado a nuevos ambientes [39]. Esto se logra mediante el uso de técnicas de interpretabilidad y visualización, como el análisis de las características importantes, las reglas de decisión, las decisiones intermedias y las

proyecciones de salida. Estas técnicas permiten a los usuarios comprender cómo el sistema IA está generando sus decisiones.

Múltiples áreas se han visto beneficiadas de modelos explicables, tales como área financiera o médica, en donde es de suma importancia poder dar una explicación de por qué se delibera una decisión. En caso particular, en el área financiera se nos perite profundizar en razones por las que es posible que un cliente se le puede ser negado un crédito. Así mismo en el área médica, en la cual es importante la opinión de múltiples expertos para validar un diagnóstico, especialmente si es inferido por modelos inteligentes [40].

Generar modelos interpretables permite entender al sistema y llevar a una corrección de sus deficiencias. Los modelos explicables permiten tener una imparcialidad a la hora de hacer la toma de decisiones, detectando consecuencias y corrigiendo sus tendencias en las bases de datos. De igual forma nos permiten hacer modelos robustos, señalando sus adversidades las cuales pueden afectar las predicciones. Los modelos interpretables nos permiten asegurarnos de que únicamente variables significativas infieran en la salida [41].

Capítulo 4. Propuesta General

En el presente trabajo, se propone la implementación de lógica difusa en algunos de los distintos puntos del flujo de datos en un sistema inteligente, con el fin de realizar una comparativa entre sistemas de clasificación y detección con métodos tradicionales y difusos.
Los métodos difusos se aplicaron en varios pasos del procesamiento de datos, incluyendo la etapa de preprocesamiento y la extracción de características. La Figura 4.1 presenta una descripción general del flujo de información en este proceso.

4.1. Fase de preprocesamiento difuso

En esta sección se describe el trabajo realizado en las pruebas de preprocesamiento difuso. Se realizó la implementación de técnicas de preprocesamiento de imágenes que implementen lógica difusa (en concreto detección de bordes) antes de pasar las imágenes a redes neuronales convolucionales para su entrenamiento. En este caso específico se planteó la utilización de detección de bordes difusos, utilizando las implementaciones con Sobel, Prewitt y Gradiente Morfológico difusos respectivamente para la experimentación. Así mismo se planteó la utilización de preprocesamientos tradicionales para realizar una comparación entre las técnicas y medir su eficiencia. La Figura 4.1 ejemplifica el flujo de datos del sistema propuesto para esta fase.

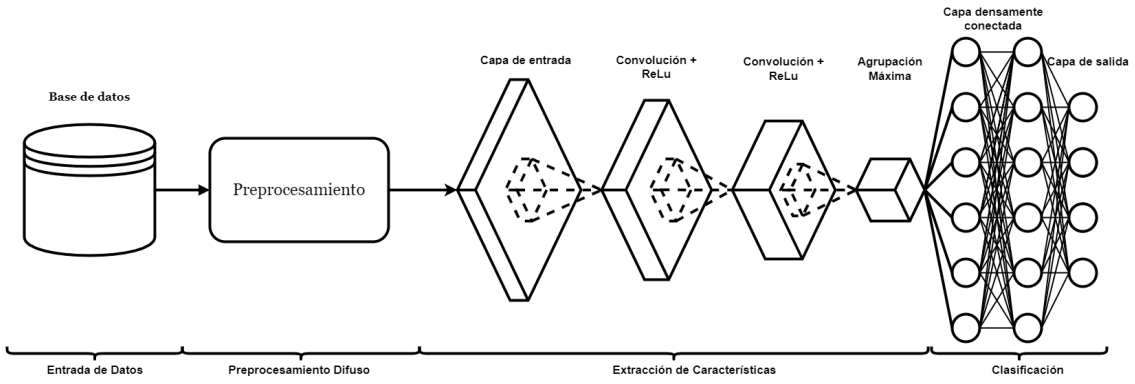

Figura. 4.1. Modelo propuesto 1: preprocesamiento difuso.

En las siguientes secciones se explica a detalle cómo se realizó la implementación de los diversos sistemas de detección de bordes difusos.

4.1.2. Detección de bordes por Prewitt y Sobel difuso

El sistema de inferencia difusa utilizado para generar el detector de bordes difusos por Prewitt y Sobel es un sistema Mamdani de tipo 1 con dos entradas, una salida y tres reglas difusas. La metodología general es definida como sigue:

- Leer la imagen de entrada. En este caso, las imágenes provienen de alguna de las bases de datos utilizadas para los sistemas de prueba.
- Obtener las entradas para el sistema de inferencia difusa. El sistema considera dos entradas, las cuales son calculadas mediante el detector de bordes de Sobel tradicional Ecuación (5) y Ecuación (6). Las variables utilizadas para este sistema son Dhorizontal y Dvertical, las cuales representan los gradientes horizontal y vertical. Las entradas son fusificadas utilizando funciones Gaussianas. La entrada Dhorizontal es granulada en las variables lingüísticas: "LowDhorizontal", "MiddleDhorizontal" y "HighDhorizontal"; de igual forma la variable Dvertical tres variables denotadas por "LowDvertical", "MiddleDvertical" y "HighDvertical". La Ecuación (36) contiene los valores para LowDhorizontal y LowDvertical, los parámetros para HighDhorizontal y HighDvertical son denotados con la Ecuación (37) y por último MiddleDhorizontal y MiddleDvertical por la Ecuación (38). Finalmente, σ es definido por la Ecuación (39) para determinar los gradientes Dhorizontal y Dvertical.

$$LowDhorizontal = min(Dhorizontal),\ LowDvertical = min(Dvertical) \tag{36}$$

$$HighDhorizontal = max(Dhorizontal),\ HighDvertical = max(Dvertical) \tag{37}$$

$$MiddleDhorizontal = (LowDhorizontal + HighDhorizontal)/4,\ MiddleDvertical = (LowDvertical + HighDvertical)/4 \tag{38}$$

$$\sigma Dhorizontal = HighDhorizontal\ /4,\ \sigma Dvertical = HighDvertical\ /4 \tag{39}$$

Una vez obtenidos los gradientes, se utilizan los parámetros para la función de membresía Gaussiana, para calcular los valores del eje horizontal (Dhorizontal) denotado por las Ecuaciones (40–42), y para el eje vertical (Dvertical) con las Ecuaciones (43–45). La Figura 4.2 contiene la visualización de estas funciones de membresía utilizadas.

$$\mu LowDhorizontal(x) = exp\left[-\frac{1}{2}\left(\frac{x - LowDhorizontal}{\sigma}\right)^2\right] \quad (40)$$

$$\mu MiddleDhorizontal(x) = exp\left[-\frac{1}{2}\left(\frac{x - MiddleDhorizontal}{\sigma}\right)^2\right] \quad (41)$$

$$\mu HighDhorizontal(x) = exp\left[-\frac{1}{2}\left(\frac{x - HighDhorizontal}{\sigma}\right)^2\right] \quad (42)$$

$$\mu LowDvertical(x) = exp\left[-\frac{1}{2}\left(\frac{x - LowDvertical}{\sigma}\right)^2\right] \quad (43)$$

$$\mu MiddleDvertical(x) = exp\left[-\frac{1}{2}\left(\frac{x - MiddleDvertical}{\sigma}\right)^2\right] \quad (44)$$

$$\mu HighDvertical(x) = exp\left[-\frac{1}{2}\left(\frac{x - HighDvertical}{\sigma}\right)^2\right] \quad (45)$$

- La salida del sistema de inferencia difusa tiene dos variables lingüísticas “Background” y “Edge”, los cuales fueron normalizados entre -4.5 y 5. Se utilizaron como centros los valores de cBackground con un valor de -5 y cEdge con valor de -4.5. El valor de σ para ambas MF se calculó con la Ecuación (46). Los parámetros para la salida (valor de cada píxel) fueron calculados utilizando las Ecuaciones (47–48), denotando el fondo y borde respectivamente. La Figura 4.3 contiene la representación de la función de membresía de salida.

$$\sigma output = abs(cBackground - cEdge)/2 \quad (46)$$

$$\mu Background(x) = exp\left[-\frac{1}{2}\left(\frac{x - (-5)}{4.75}\right)^2\right] \quad (47)$$

$$\mu Edge(x) = exp\left[-\frac{1}{2}\left(\frac{x-4.5}{4.75}\right)^2\right] \tag{48}$$

Las reglas del sistema son denotadas en la Tabla 4.1, estas fueron basadas en conocimiento de expertos, probadas en casos de estudio previos.

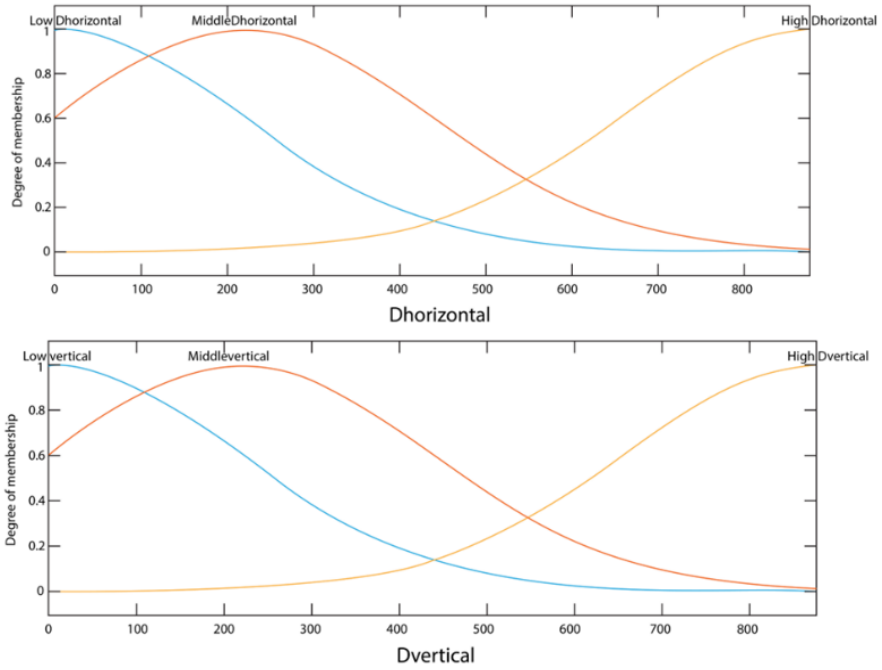

Figura. 4.2. Funciones de membresía Dhorizontal y Dvertical utilizadas en Prewitt y Sobel difuso.

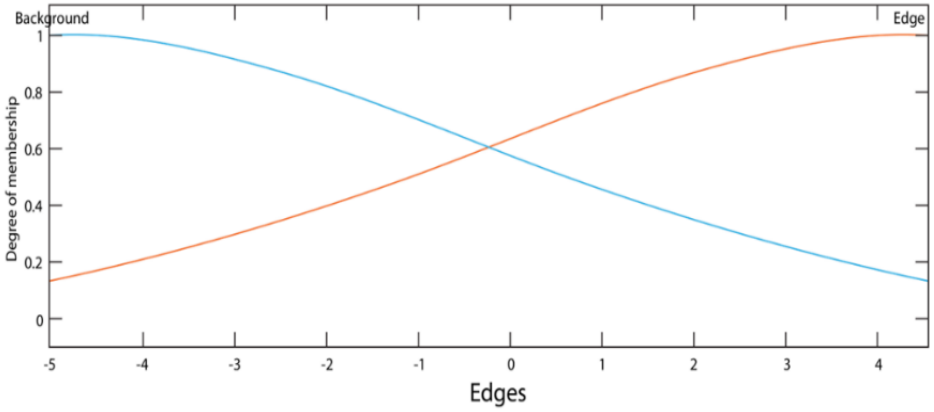

Figura. 4.3. Funciones de membresía para obtener bordes difusos.

Tabla 4.1. Reglas para detector de bordes difusos por Prewitt y Sobel.

1	If (Dhorizontal is HighDhorizontal) or (Dvertical is HighDvertical) then (Edges is Edge)
2	If (Dhorizontal is MiddleDhorizontal) or (Dvertical is MiddleDvertical) then (Edges is Edge)
3	If (Dhorizontal is LowDhorizontal) and (Dvertical is LowDvertical) then (Edges is Background)

4.1.3. Detección de bordes por gradiente morfológico difuso

Similar al apartado anterior, se realizó la implementación de un sistema de inferencia difusa Mamdani tipo 1 para realizar el cálculo de los bordes. En este caso se tienen cuatro entradas (correspondientes a cada gradiente) y dos salidas (para bordo y fondo, similar al sistema de Sobel difuso). Los pasos para realizar la implementación son los siguientes.

- Obtención de los cuatro gradientes de la imagen. Los gradientes G_1, G_2, G_3, G_4 calculados con la Ecuación (8), son utilizados como entrada para el sistema de inferencia difusa.
- Definir variables lingüísticas para el sistema de inferencia difusa. Las cuatro entradas son separadas con tres funciones de membresía Gaussianas, utilizando los términos lingüísticos "Low", "Middle" y "High" para cada gradiente, estos son denotados por las Ecuaciones (49–52). El σ de cada gradiente es calculado utilizando la Ecuación (53).

$$Grad1Low = \min(Grad1)\,, Grad2Low = \min(Grad2), Grad3Low = \min(Grad3)\,, Grad4Low = \min\,(Grad4) \tag{49}$$

$$Grad1High = max(Grad1)\,, Grad2High = max(Grad2), Grad3High = max(Grad3)\,, Grad4High = max\,(Grad4) \tag{50}$$

$$Grad1Middle = \ (Grad1Low + \ Grad1High)/2, Grad2Middle = (Grad2Low + \ Grad2High)/2 \tag{51}$$

$$Grad3Middle = \ (Grad3Low + \ Grad3High)/2, Grad4Middle = (Grad4Low + \ Grad4High)/2 \tag{52}$$

$$\sigma Grad1 = Grad1High/4, \sigma Grad2 = Grad2High/4, \sigma Grad3 = Grad3High/4, \sigma Grad4 = Grad4High/4 \quad (53)$$

Las funciones de Membresía Gaussianas son calculadas utilizando las Ecuaciones (54–65) para cada gradiente respectivamente. La Figura 4.4 contiene la visualización de las funciones de membresía para cada gradiente.

$$\mu Grad1Low(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad1Low}{\sigma Grad1}\right)^2\right] \quad (54)$$

$$\mu Grad1Middle(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad1Middle}{\sigma Grad1}\right)^2\right] \quad (55)$$

$$\mu Grad1High(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad1High}{\sigma Grad1}\right)^2\right] \quad (56)$$

$$\mu Grad2Low(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad2Low}{\sigma Grad2}\right)^2\right] \quad (57)$$

$$\mu Grad2Middle(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad2Middle}{\sigma Grad2}\right)^2\right] \quad (58)$$

$$\mu Grad2High(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad2High}{\sigma Grad2}\right)^2\right] \quad (59)$$

$$\mu Grad3Low(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad3Low}{\sigma Grad3}\right)^2\right] \quad (60)$$

$$\mu Grad3Middle(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad3Middle}{\sigma Grad3}\right)^2\right] \quad (61)$$

$$\mu Grad3High(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad3High}{\sigma Grad3}\right)^2\right] \quad (62)$$

$$\mu Grad4Low(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad4Low}{\sigma Grad4}\right)^2\right] \quad (63)$$

$$\mu Grad4Middle(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad4Middle}{\sigma Grad4}\right)^2\right] \quad (64)$$

$$\mu Grad4High(x) = exp\left[-\frac{1}{2}\left(\frac{x - Grad4High}{\sigma Grad4}\right)^2\right] \quad (65)$$

- Cálculo de la salida. La salida se encuentra separada con dos valores lingüísticos "Background" y "Edge"; similar al sistema implementado para calcular bordes por Sobel. Los centros de este sistema se encuentran normalizados entre -5 y 4.5. El centro del valor para Background se encuentra representado con la variable $cBackground = -5$ y para Edge con $cEdge = 4.5$. El valor de σ para ambas variables fue calculado con la Ecuación (66).

$$\sigma output = abs(cBackground - cEdge)/2 \tag{66}$$

  Los parámetros para la salida de la función de Membresía (Bordes) son expresados por las Ecuaciones (67–68).

$$\mu Background(x) = exp\left[-\frac{1}{2}\left(\frac{x - \text{cBackground}}{\sigma\text{output}}\right)^2\right] \tag{67}$$

$$\mu Edge(x) = exp\left[-\frac{1}{2}\left(\frac{x - \text{cEdge}}{\sigma\text{output}}\right)^2\right] \tag{68}$$

- Reglas del sistema de inferencia difusa Mamdani. La base de datos de reglas se encuentra expresado en la Tabla 4.2.

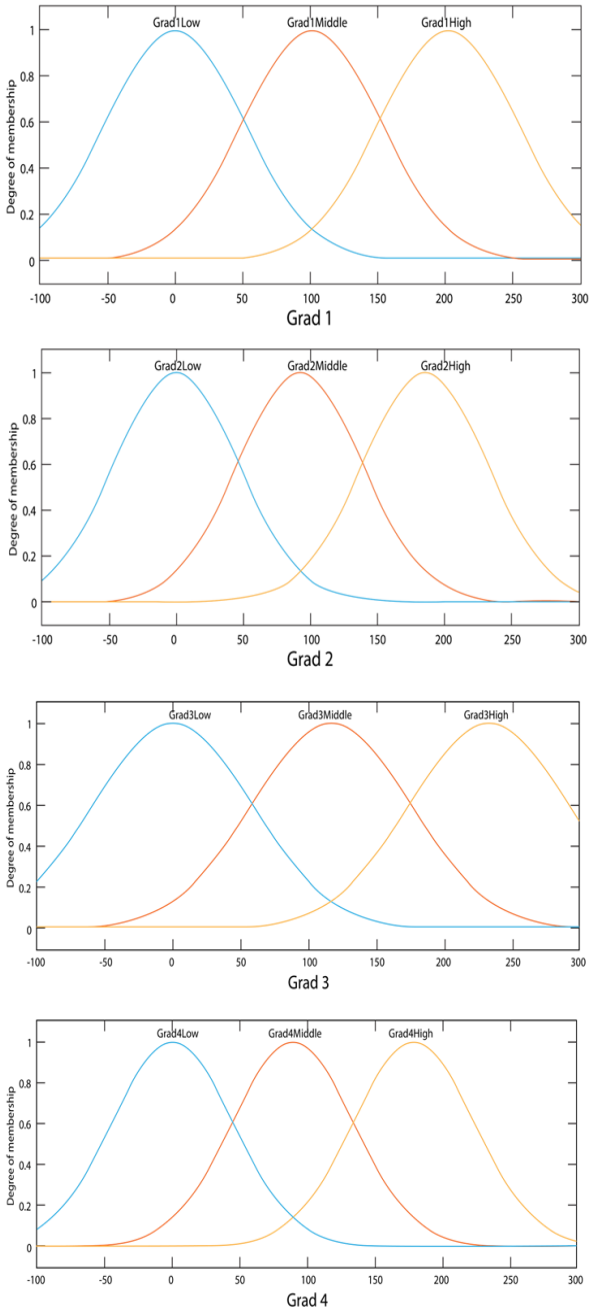

Figura. 4.4. Funciones de membresía utilizadas en gradiente morfológico difuso.

Tabla 4.2. Reglas para detector de bordes difusos por gradiente morfológico.

1	If (Grad1 is Grad1High) or (Grad2 is Grad2High) or (Grad3 is Grad3High) or (Grad4 is Grad4High) then (Edges is Edge)
2	If (Grad1 is Grad1Middle) or (Grad2 is Grad2Middle) or (Grad3 is Grad3Middle) or (Grad4 is Grad4Middle) then (Edges is Edge)
3	If (Grad1 is Grad1Low) and (Grad2 is Grad2Low) and (Grad3 is Grad3Low) and (Grad4 is Grad4Low) then (Edges is Background)

En el Algoritmo 1 se ejemplifica el funcionamiento de un sistema para cálculo de bordes difusos por método de Prewitt o Sobel. En la Figura 4.5, se ilustra la detección de bordes aplicando el método Sobel difuso.

Algoritmo 1. *Procesamiento de bordes difusos*

```
Seleccionar el operador adecuado para calcular los gradientes.
Caso Prewitt:
     Utilizar filtro Prewitt Ecuaciones (6)-(7).
Caso Sobel:
     Utilizar filtro Sobel
Leer la imagen de entrada f
Obtener las dimensiones
[ren, col] = tamaño(f)
Calcular los gradientes clásicos correspondientes al filtro difuso
deseado.
[Dx, Dy] = zeros(ren, col) // Generar una matriz de ceros con las mismas
dimensiones que f para capturar los gradientes.
para i = 0 en ren:
     para j = 0 en col:
          Dx[i,j] = sum(* f [i: i+3, j: j+3])
          Dy[i,j] = sum(* f [i: i+3, j: j+3])
     fin para
fin para
Generar el controlador difuso requerido utilizando las reglas de la
Tabla 4.1
Fusificar las entradas de los dos gradientes Dx y Dy, utilizando la
función de membresía Gaussiana
Inferir los bordes con el control difuso seleccionado.
Defusificar la salida (bordes) del controlador.
```

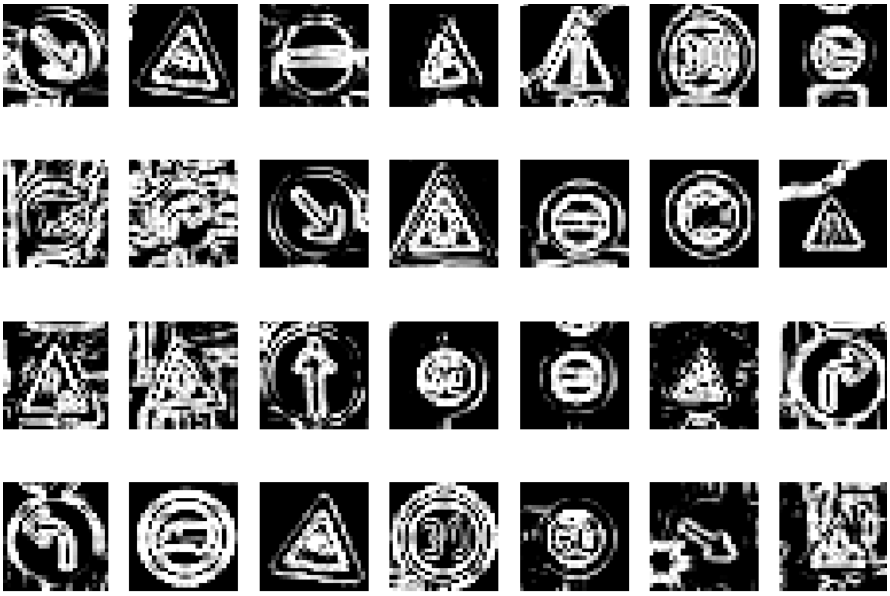

Figura. 4.5. Detección de bordes por método de Sobel difuso aplicado a base de datos GTSRB [42].

4.2. Fase de convolución difusa

A continuación, se presenta una alteración de la arquitectura propuesta de una red neuronal convolucional para trabajar en el espectro difuso.

La modificación propuesta consta en convertir los datos utilizados al realizar el proceso de convolución en una CNN tradicional a difusos utilizando funciones de Membresía para la entrada (base de datos), así como los filtros aleatorios. La Figura 4.3 muestra un diagrama simplificado de este proceso.

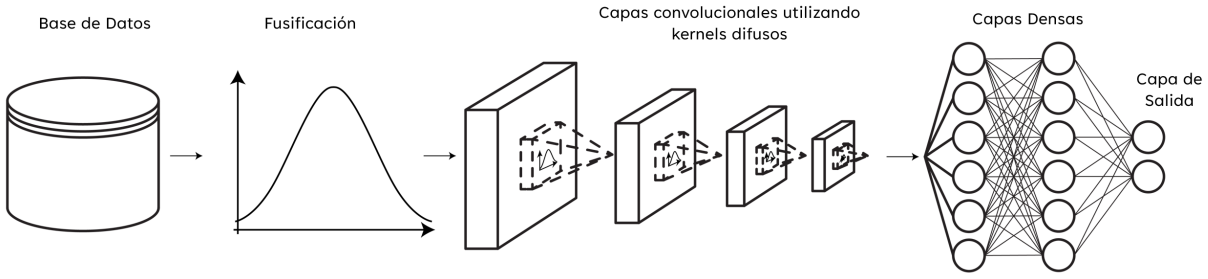

Figura. 4.6. Modelo propuesto 2: fusificación de entradas y filtros.

El proceso simplificado consta de los siguientes pasos:

1. Fusificación de base de datos.

2. Fusificación de filtros de capas de convolución.
3. Proceso de convolución (aplicando filtros difusos a imágenes difusas) con Rectificación Lineal Unitaria (ReLu).
4. Agrupamiento.
5. Defusificación de tensores.
6. Capa densamente conectada.
7. Capa de salida.

Igual que en la etapa de preprocesamiento de los datos, se aplicaron y entrenaron los cuatro modelos propuestos utilizando funciones de membresía en los datos de entrada y en los filtros aleatorios. Los parámetros de entrenamiento son los mismos que en el paso anterior. Para fusificación los kernels y los datos, se propuso experimentar con tres funciones de membresía diferentes.

- Función de membresía Triangular Ecuación (23).
- Función de membresía Gaussiana Ecuación (25).
- Función de membresía S Ecuación (26).

Los rangos de las funciones de membresía fueron denotados por las siguientes restricciones: Los datos de entrenamiento (imágenes en escala de grises) con rango entre [0,255]. Los filtros aleatorios (Originalmente generados con inicialización de Xavier) generados de una distribución normal con rango entre [-1,1]. La Figura 4.4 contiene la visualización de las funciones de membresía utilizadas para el proceso de convolución difusa. La implementación de estas funciones de membresía fue realizada en Python utilizando la librería de Scikit-Fuzzy.

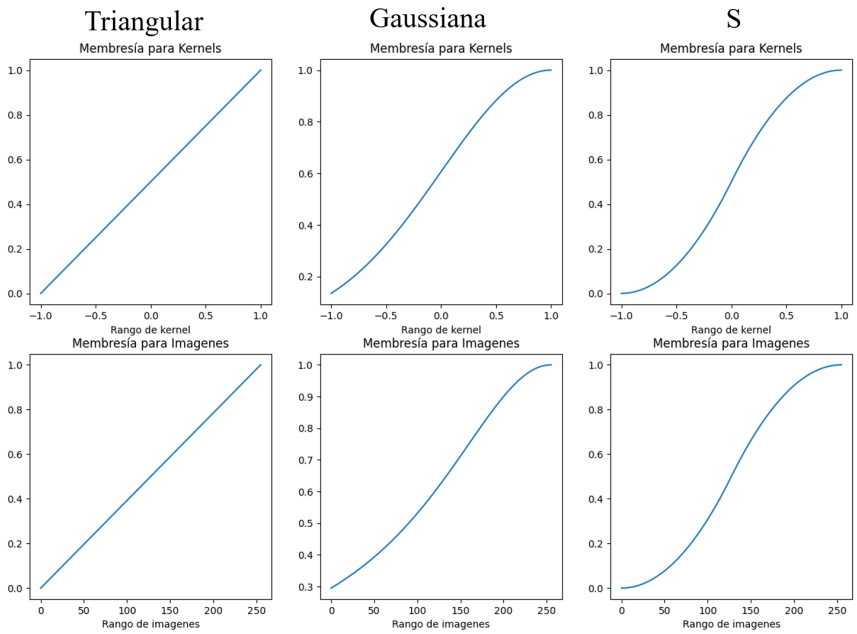

Figura. 4.7. Funciones de membresía utilizadas para proceso de convolución difusa.

4.3. Fase de composición difusa

En esta fase de la propuesta se planteó ampliar el espectro de utilización de lógica difusa en las capas de extracción de características. Para esto, se planteó en alterar la operación de composición, sustituyendo el producto escalar realizado por una composición matricial entre los filtros y la ventana. En caso concreto el filtrado de las características es realizado por una composición Min-Max utilizando los filtros (kernels) e información el espectro difuso. La Figura 4.4 ejemplifica el flujo de datos utilizado para realizar el proceso de composición difuso.

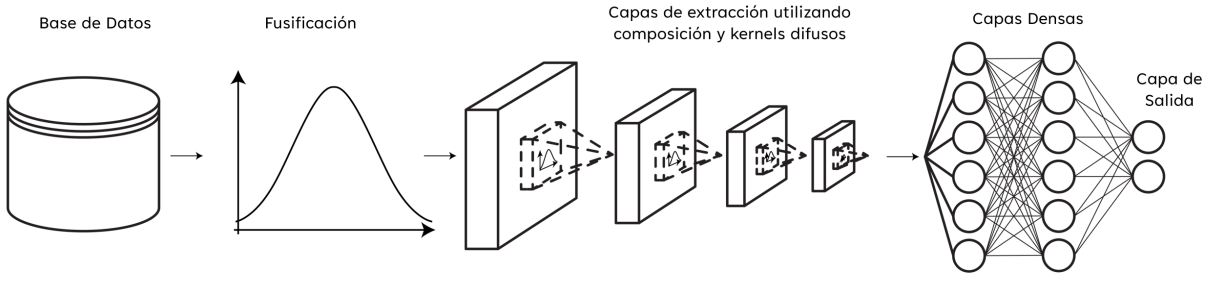

Figura. 4.8. Arquitectura de CNN utilizando composición para extracción de características.

Dado a que el proceso de la capa de composición difusa se encuentra inspirado en una capa de convolución tradicional, donde el producto obtenido por los filtros y la sub-ventana de la imagen es sustituido por la operación de composición de matrices aplicado en sistemas de control de lógica difusa. Debido a que las entradas y los filtros se encuentran fusificados, es posible tratarlos como conjuntos difusos, dado a que $I \circ K = \{[(x,y),\ min\, max(\mu_I(x,y), \mu_k(y,x))] |\ x \in X,\ y \in Y\}$, donde I representa a la imagen de entrada y K al conjunto de filtros, por lo cual es posible derivar la Ecuación (69), donde:

- I: Imagen de entrada.
- K: kernel (filtro) de entrada.
- i: columna de la imagen.
- j: renglón de la imagen.
- r: renglón del filtro.
- c: columna del filtro.
- $(\mathrm{I} \circ K)$: imagen resultante.

$$\mu_{I \circ K}[i,j] = Max\left[\sum_{r=-n}^{n}\sum_{c=-m}^{m} Min\left[Max\left[k[r,c], I[i-r, j-c]\right]\right]\right] \tag{69}$$

El Algoritmo 2 detalla el proceso denotado como composición difusa. Los resultados de la implementación del algoritmo se ilustran en la Figura 4.9, donde se comparan los efectos de la composición difusa contra los de la convolución clásica, utilizando 1, 4 y 8 filtros en la tradicional imagen de referencia Lena. La entrada del algoritmo propuesto conforma de dos tensores de 4 dimensiones, el primer tensor X, consta de las imágenes de entrenamiento utilizada para alimentar el modelo, la forma del filtro se encuentra compuesto por los siguientes elementos, $\mathrm{X} = [batch,\ ancho,\ alto,\ dimenciones]$. El segundo tensor de entrada denominado $filtros$ consta de las matrices pequeñas utilizadas para realizar el proceso de extracción de características (similares a los filtros aleatorios utilizados en el proceso de convolución clásica). Este filtro de igual forma consta de 4 dimensiones, la forma del filtro está compuesto por los siguientes elementos,$filtros = [filtro\ ancho, filtro\ alto,$ $filtro\ dimenciones\ entrada,\ filtro\ \ dimensiones\ salida]$. La salida del algoritmo consta de igual forma un tensor de 4 dimensiones el cual sus elementos están compuestos por la relación

al tensor de entrada sustituyendo las dimensiones por la dimensión de filtros de salida. $Salida = [batch,\ ancho,\ alto,\ filtro\ dimensiones\ salida]$.

Algoritmo 2. *Proceso de composición difusa.*

```
Salida []
para b en batch:
    para x en ancho imagen:
        para y en alto imagen:
            para k_di en fil dim entrada:
                para k_do en fil dim salida:
                    para k_x en fil ancho:
                        para k_y en fil alto:
                            Salida[b,x,y,k_do] =
                               Max(Min(Max(X[b,x,y,k_di],
                               filtro[k_x,k_y,ki_di,k_do])));
                        fin para
                    fin para
                fin para
            fin para
        fin para
    fin para
fin para
```

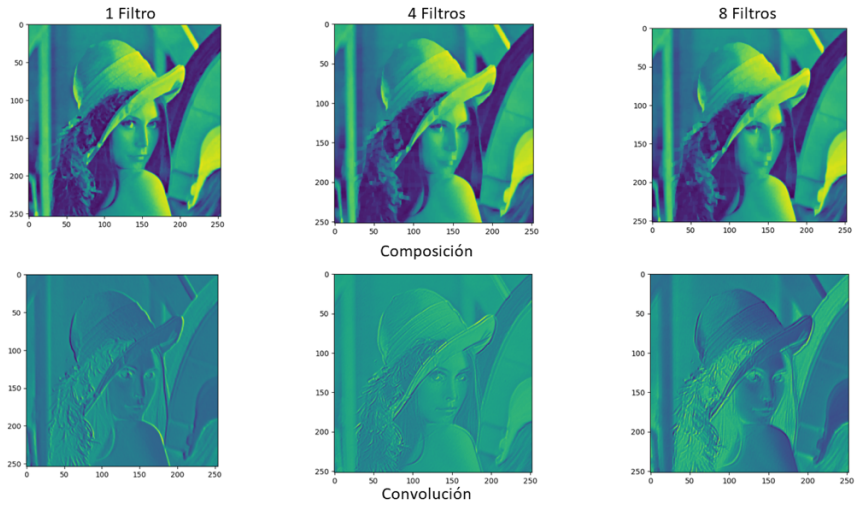

Figura. 4.9. Ejemplo de composición difusa comparado contra convolución utilizando distinto número de filtros.

Capítulo 5. Experimentos y Resultados

El siguiente capítulo contiene la experimentación realizada a distintas bases de datos con los modelos propuestos. Las bases de datos fueron procesadas con distintas metodologías de preprocesamiento de imágenes descritas en capítulo 4, antes de que pasaran a redes neuronales convolucionales para su entrenamiento. Para la tarea de clasificación de objetos se realizaron pruebas con tres bases de datos de referencia relacionadas al área de clasificación de bases de datos, tales como la base de datos GTSRB, base de datos BELGIUMTS [43] y la base de datos CTSD [44].

La implementación y entrenamiento de los modelos fue realizada en el lenguaje de programación Python en su versión 3.8 utilizando la librería de TensorFlow [45] con el API de Keras. El desarrollo fue generado utilizando la herramienta de contenedores virtuales de NVIDIA Container Toolkit para realizar un contenedor acelerado con GPU el cual se ejecutó en un dispositivo con las siguientes especificaciones: Sistema operativo Pop!_OS en la versión 20.04 LTS, Intel Core I7 12700HQ, 64 GB de RAM a 3400 MHz, Tarjeta gráfica Nvidia RTX 3080 Mobile con 16 GB de memoria de video DDR6 y RTX 3070Ti con 8GB de memoria de video DDR6X utilizada con una adaptador eGPU Razer Core X sobre Thunderbolt 3 y 2 TB de disco de estado sólido para almacenamiento.

5.1. Bases de datos utilizadas

En esta sección, se presenta las distintas bases de datos utilizadas para desarrollar las propuestas de la presente investigación, se incluyen aspectos importantes, tales como su origen, estructura y contenido. Las bases de datos fueron seleccionadas con el objetivo de tener información relevante en el área de los objetivos específicos de la investigación.

5.1.1. Base de datos GTSRB [42]

La German Traffic Sign Recognition Benchmark (GTSRB) contiene múltiples clases de señales de tránsito utilizado en la competencia IJCNN en 2011 y publicado en el 2013. Las imágenes

fueron extraídas de cuadros de video tomado por el Institut Für Neuro Informatik. El set de datos se encuentra desbalanceado, mientras que las proporciones de los objetos de interés son constantes. Contiene 51,839 imágenes con 43 clases distintas. La GTSRB se encuentra separada en sets de entrenamiento con 39,209 imágenes y el set de prueba con 12,630 imágenes. La base de datos se encuentra en el formato PPM con dimensiones de 28px × 28px. La Figura 5.1 contiene una muestra de algunas de las imágenes encontradas en el conjunto de datos.

Figura. 5.1. Muestra Base de datos GTSRB.

5.1.2. Base de datos BELGIUMTS [43]

La Belgium Traffic Sign Classification Benchmark (BELGIUMTS), es una base de datos proveniente de Bélgica, la cual contiene imágenes de distintas señales de tránsito encontradas en el país. Contiene 62 categorías distintas las cuales fueron recortadas de cuadros de video. Comparada contra GTSRB, esta es significativamente más pequeña con únicamente 7,095 imágenes, las que de igual forma se encuentran separadas en sets de entrenamiento y pruebas, con 4,575 imágenes y 2,520 imagen respectivamente. La BelgiumTS se encuentra en formato PPM con una resolución de 32px × 32px. La Figura 5.2 contiene una muestra de la base de datos.

Figura. 5.2. Muestra base de datos BELGIUMTS.

5.1.3. Base de datos CTSD [44]

La CTSD (Traffic Sign Recognition Database) es una base de datos proveniente de China, con 58 categorías distintas de señalamientos, donde en cada imagen contiene únicamente una señal de tránsito. La base de datos incluye un total de 6,164 imágenes, donde 4,170 imágenes pertenecen al conjunto de entrenamiento y 1994 imágenes del conjunto de prueba. Los datos se encuentran en formato PPM con una resolución de 64px × 64px. La Figura 5.3 ejemplifica el contenido de la base de datos.

Figura. 5.3. Muestra base de datos CTSD.

5.2. Modelos propuestos para detección

Se propusieron cuatro arquitecturas de redes neuronales convolucionales para realizar el área de clasificación. Todas las arquitecturas fueron entrenadas con las distintas técnicas de preprocesamiento e hiper parámetros a excepción del tamaño de lote (batch) el cual fue variado dependiendo del tamaño de la base de datos. Las Tablas 5.1 a 5.3 muestran las arquitecturas utilizadas para el entrenamiento para los modelos CNN-I a CNN-IV.

Tabla 5.1. Arquitectura Modelo CNN-I.

Tipo de capa	Neuronas/Filtros	Tamaño de filtro
Entrada Convolución + ReLu	64	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	64	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	128	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	128	3x3
Agrupación Máxima	--	2x2
Expulsión	--	0.75
Aplanamiento	--	--
Densamente conectada	512	--
Salida	N-salidas	--

Tabla 5.2. Arquitectura Modelo CNN-II

Tipo de capa	Neuronas/Filtros	Tamaño de filtro
Entrada Convolución + ReLu	64	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	128	3x3
Agrupación Máxima	--	2x2
Expulsión	--	0.75
Aplanamiento	--	--
Densamente conectada	512	--
Salida	N-salidas	--

Tabla 5.3. Arquitectura Modelo CNN-III

Tipo de capa	Neuronas/Filtros	Tamaño de filtro
Entrada Convolución + ReLu	64	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	64	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	128	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	128	3x3
Agrupación Máxima	--	2x2
Expulsión	--	0.45
Aplanamiento	--	--
Densamente conectada	1024	--
Densamente conectada	512	--
Salida	N-salidas	--

Tabla 5.4. Arquitectura Modelo CNN-IV

Tipo de capa	Neuronas/Filtros	Tamaño de filtro
Entrada Convolución + ReLu	32	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	64	3x3
Agrupación Máxima	--	2x2
Convolución + ReLu	128	3x3
Agrupación Máxima	--	2x2
Expulsión	--	0.35
Aplanamiento	--	--
Densamente conectada	256	--
Salida	N-salidas	--

5.3. Resultados de preprocesamiento de imágenes difusas

Los cuatro modelos propuestos en la sección anterior fueron entrenados con los siguientes parámetros:

- Épocas: 100
- Función de pérdida: Entropía cruzada categórica dispersa.
- Función de optimización: Adam (SGD optimizado), parámetros predeterminados de Keras.
    - Épsilon: $1e^{-7}$
    - Beta 1: 0.9
    - Beta 2: 0.999
    - Paso de aprendizaje: 0.001
- Tamaño de lote (batch): variante.
- Se realizaron con un total de 30 experimentos independientes por cada tipo de preprocesamiento.

El preprocesamiento de imágenes clásico fue realizado utilizando la librería de OpenCV [46] con el API de Python, el preprocesamiento difuso se realizó utilizando la herramienta de Fuzzy Logic Toolbox en Matlab debido a su eficiencia.

Esta sección contiene los resultados de los entrenamientos realizados a las distintas bases de datos, se entrenaron los cuatro modelos propuestos en las Tablas 5.1 a 5.4, realizando 30 experimentos independientes por cada tipo de preprocesamiento. Los preprocesamientos aplicados a las imágenes constan de los siguientes:

- Color.
- Escala de grises.
- Gradiente morfológico.
- Prewitt.
- Sobel.
- Gradiente morfológico difuso.
- Sobel difuso.
- Sobel a color difuso.
- Prewitt difuso.

En conjunto a los resultados de los entrenamientos se incluye un análisis estadístico comparativo para determinar la existencia de diferencia significativa al momento de utilizar un método de preprocesamiento en específico. Para esta sección únicamente se comparan los dos mejores métodos de preprocesamiento para cada modelo aplicado a la base de datos. Los parámetros de la prueba estadística realizada son los siguientes:

- Prueba de cola derecha.
- $\alpha = 0.05$ (Nivel de confidencia del 95%, zona de rechazo $z_c = 1.96$).
- $n = 30$.
- H_0 : La utilización de una técnica de preprocesamiento (μ_1) ofrece menor o igual precisión que la utilización de imágenes a color. (μ_2). $H_0: \mu_1 \leq \mu_2$.
- H_a: La utilización de una técnica de preprocesamiento (μ_1) ofrece mayor precisión que la utilización de imágenes a color (μ_2). $H_a: \mu_1 > \mu_2$ (afirmación).

5.3.1. Base de datos GTSRB

En esta sección se encuentran los resultados del entrenamiento con la base de datos GTSRB, la Tabla 5.5 contiene resultados del estado del arte, donde diversos modelos fueron aplicados en esta base de datos, que serán utilizados como referencia y para realizar un análisis comparativo. Las Tablas 5.6 a 5.9 plasman los resultados obtenidos al aplicar los modelos propuestos CNN-I a CNN-IV, el análisis estadístico incluido donde se prueban los dos mejores métodos de preprocesamiento para cada modelo se encuentra plasmado en la Tabla 5.10.

Tabla 5.5. Estado del Arte aplicado a base de datos GTSRB.

No.	Metodología	Precisión
1	GDBM [47]	0.9934
2	HLSGD [48]	0.9965
3	OneCNN [49]	0.9911
4	Single CNN with 3 STNs [50]	0.9971

Tabla 5.6. Resultado de entrenamiento CNN-I aplicado a base de datos GTSRB

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9209	0.9739	0.9650	0.0102
2	Escala de Grises	0.9419	0.9738	0.9656	0.0080
3	Gradiente Morfológico	0.6732	0.6948	0.6831	0.0054
4	Prewitt	0.9458	0.9577	0.9521	0.0038
5	Sobel	0.9356	0.9544	0.9460	0.0044
6	Grad. Morf. Difuso	0.7253	0.7476	0.7337	0.0043
7	Sobel Difuso	0.8707	0.8818	0.8768	0.0029
8	Sobel Color Difuso	0.8114	0.8304	0.8209	0.0046
9	Prewitt Difuso	0.8709	0.8837	0.8773	0.0029

Tabla 5.7. Resultado de entrenamiento CNN-II aplicado a base de datos GTSRB.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9391	0.9558	0.9495	0.0048
2	Escala de Grises	0.9458	0.9614	0.9565	0.0035
3	Gradiente Morfológico	0.6732	0.6948	0.6831	0.0054
4	Prewitt	0.9353	0.9464	0.9412	0.0029
5	Sobel	0.9338	0.9432	0.9393	0.0024
6	Grad. Morf. Difuso	0.7304	0.7427	0.7356	0.0028
7	Sobel Difuso	0.8584	0.8743	0.8675	0.0036
8	Sobel Color Difuso	0.8021	0.8162	0.8082	0.0031
9	Prewitt Difuso	0.8709	0.8837	0.8773	0.0029

Tabla 5.8. Resultado de entrenamiento CNN-III aplicado a base de datos GTSRB.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9470	0.9723	0.9626	0.0060
2	Escala de Grises	0.9457	0.9705	0.9593	0.0065
3	Gradiente Morfológico	0.6732	0.6948	0.6831	0.0054
4	Prewitt	0.9367	0.9539	0.9469	0.0038
5	Sobel	0.9285	0.9488	0.9413	0.0046
6	Grad. Morf. Difuso	0.7209	0.7426	0.7337	0.0048
7	Sobel Difuso	0.8646	0.8802	0.8723	0.0034
8	Sobel Color Difuso	0.8049	0.8238	0.8180	0.0046
9	Prewitt Difuso	0.8709	0.8837	0.8773	0.0029

Tabla 5.9. Resultado de entrenamiento CNN-IV aplicado a base de datos GTSRB.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9462	0.9689	0.9603	0.0059
2	Escala de Grises	0.9495	0.9671	0.9585	0.0046
3	Gradiente Morfológico	0.6732	0.6948	0.6831	0.0054
4	Prewitt	0.9315	0.9512	0.9424	0.0041
5	Sobel	0.9317	0.9482	0.9401	0.0037
6	Grad. Morf. Difuso	0.7253	0.7476	0.7337	0.0043
7	Sobel Difuso	0.8707	0.8818	0.8768	0.0029
8	Sobel Color Difuso	0.8114	0.8304	0.8209	0.0046
9	Prewitt Difuso	0.8709	0.8837	0.8773	0.0029

5.3.1.2. Análisis estadístico para base de datos GTSRB

Para determinar qué tipo de preprocesamiento es el mejor para este caso de estudio, se decidió evaluar los experimentos con una prueba Z de cola derecha con 95% de significancia. Se planteo comparar los dos tipos de preprocesamiento con los porcentajes de precisión más altos para cada modelo. En este caso fue Escala de grises y las imágenes a color para los 4 modelos propuestos.

Tabla 5.10. Análisis estadístico para comparar tipos de preprocesamiento.

Modelo	Preprocesamiento	$\overline{x_1}$	σ_1	$\overline{x_2}$	σ_2	Valor-Z
CNN-I	Color vs. Escala de grises	0.9656	0.0080	0.9650	0.0102	0.2535
CNN-II	Color vs. Escala de grises	0.9565	0.0035	0.9495	0.0048	**6.4541**
CNN-III	Color vs. Escala de grises	0.9593	0.0065	0.9626	0.0060	-2.0433
CNN-IV	Color vs. Escala de grises	0.9585	0.0046	0.9603	0.0059	-1.3178

5.3.2. Base de datos BELGIUMTS

La experimentación con la base de datos BelgiumTS se encuentra detallada en esta sección. Las aplicaciones del estado del arte donde se trabaja con esta base de datos se encuentran detallados en la Tabla 5.11. Los resultados de los entrenamientos aplicados a BelgiumTS se encuentran plasmados en las Tablas 5.12 a 5.15 para cada una de las arquitecturas propuestas. Los resultados de la prueba estadística comparando los preprocesamientos en esta base de datos se encuentran en la Tabla 5.16.

Tabla 5.11. Estado del Arte aplicado a base de datos BelgiumTS.

No.	Metodología	Precisión
1	GDBM [47]	0.9892
2	OneCNN [49]	0.9817
3	Single CNN with 3 STNs [50]	0.9887
4	VGG-16 con GA optimizada [51]	0.9916

Tabla 5.12. Resultado de entrenamiento CNN-I aplicado a base de datos BELGIUMTS.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.8905	0.9750	0.9574	0.0163
2	Escala de Grises	0.9230	0.9746	0.9567	0.0108
3	Gradiente Morfológico	0.7056	0.8929	0.8598	0.0328
4	Prewitt	0.9202	0.9679	0.9521	0.0107
5	Sobel	0.9266	0.9532	0.9391	0.0067
6	Grad. Morf. Difuso	0.8468	0.8742	0.8592	0.0075
7	Sobel Difuso	0.9135	0.9492	0.9375	0.0077
8	Sobel Color Difuso	0.8730	0.9290	0.9002	0.0109
9	Prewitt Difuso	0.9048	0.9532	0.9339	0.0106

Tabla 5.13. Resultado de entrenamiento CNN-II aplicado a base de datos BELGIUMTS.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9032	0.9639	0.9394	0.0128
2	Escala de Grises	0.9135	0.9456	0.9319	0.0081
3	Gradiente Morfológico	0.7567	0.8802	0.8487	0.0235
4	Prewitt	0.9052	0.9619	0.9393	0.0128
5	Sobel	0.9000	0.9321	0.9192	0.0078
6	Grad. Morf. Difuso	0.8377	0.8746	0.8538	0.0068
7	Sobel Difuso	0.9060	0.9369	0.9228	0.0083
8	Sobel Color Difuso	0.8746	0.8988	0.8885	0.0064
9	Prewitt Difuso	0.9091	0.9397	0.9261	0.0068

Tabla 5.14. Resultado de entrenamiento CNN-III aplicado a base de datos BELGIUMTS.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9119	0.9702	0.9522	0.0147
2	Escala de Grises	0.8873	0.9639	0.9396	0.0186
3	Gradiente Morfológico	0.7754	0.8734	0.8401	0.0241
4	Prewitt	0.9012	0.9611	0.9391	0.0158
5	Sobel	0.8901	0.9472	0.9251	0.0133
6	Grad. Morf. Difuso	0.7718	0.8563	0.8351	0.0162
7	Sobel Difuso	0.8770	0.9464	0.9198	0.0138
8	Sobel Color Difuso	0.8508	0.9147	0.8877	0.0128
9	Prewitt Difuso	0.8813	0.9429	0.9245	0.0150

Tabla 5.15. Resultado de entrenamiento CNN-IV aplicado a base de datos BELGIUMTS.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.9159	0.9651	0.9453	0.0135
2	Escala de Grises	0.9028	0.9583	0.9341	0.0159
3	Gradiente Morfológico	0.7302	0.8714	0.8344	0.0297
4	Prewitt	0.8869	0.9488	0.9306	0.0160
5	Sobel	0.8587	0.9421	0.9207	0.0180
6	Grad. Morf. Difuso	0.8103	0.8571	0.8330	0.0111
7	Sobel Difuso	0.8921	0.9302	0.9178	0.0089
8	Sobel Color Difuso	0.8373	0.8956	0.8731	0.0140
9	Prewitt Difuso	0.8956	0.9433	0.9144	0.0121

5.3.2.2. Análisis estadístico para base de datos BELGIUMTS

Se realizó una prueba Z con 95% de significancia para comparar los dos preprocesamientos con mayor promedio de precisión, similarmente a la GTSRB se obtuvo que Escala de grises y las imágenes a Color fueron los mejores promedios.

Tabla 5.16. Análisis estadístico para comparar tipos de preprocesamiento con base de datos BELGIUMTS.

Modelo	Preprocesamiento	$\overline{x_1}$	σ_1	$\overline{x_2}$	σ_2	Valor-Z
CNN-I	Color vs. Escala de grises	0.9567	0.0108	0.9574	0.0163	-0.1961
CNN-II	Color vs. Escala de grises	0.9319	0.0081	0.9394	0.0128	-2.7119
CNN-III	Color vs. Escala de grises	0.9396	0.0186	0.9522	0.0147	-2.911
CNN-IV	Color vs. Escala de grises	0.9341	0.0159	0.9453	0.0135	-2.9411

5.3.3. Base de datos CTSD

Los resultados de la experimentación realizada para CTSD son mostrados en las Tablas 5.17 a 5.20, y los resultados de la prueba estadística se encuentran en la Tabla 5.21.

Tabla 5.17. Resultado de entrenamiento CNN-I aplicado a base de datos CTSD.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.4955	0.7282	0.6684	0.0492
2	Escala de Grises	0.6189	0.7232	0.6674	0.0290
3	Gradiente Morfológico	0.5095	0.6219	0.5698	0.0268
4	Prewitt	0.5878	0.7242	0.6855	0.0310
5	Sobel	0.6319	0.7422	0.6863	0.0262
6	Grad. Morf. Difuso	0.5807	0.6830	0.6364	0.0260
7	Sobel Difuso	0.6620	0.7603	0.7114	0.0263
8	Sobel Color Difuso	0.5838	0.7172	0.6756	0.0293
9	Prewitt Difuso	0.6319	0.7382	0.6985	0.0257

Tabla 5.18. Resultado de entrenamiento CNN-II aplicado a base de datos CTSD.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.4062	0.5858	0.5318	0.0370
2	Escala de Grises	0.4393	0.5707	0.5068	0.0319
3	Gradiente Morfológico	0.3691	0.4885	0.4532	0.0267
4	Prewitt	0.4955	0.5878	0.5367	0.0229
5	Sobel	0.5286	0.6169	0.5599	0.0218
6	Grad. Morf. Difuso	0.4333	0.5597	0.4947	0.0281
7	Sobel Difuso	0.5045	0.5988	0.5473	0.0213
8	Sobel Color Difuso	0.4694	0.5998	0.5376	0.0233
9	Prewitt Difuso	0.4774	0.5908	0.5333	0.0252

Tabla 5.19. Resultado de entrenamiento CNN-III aplicado a base de datos CTSD.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.4885	0.7081	0.6356	0.0565
2	Escala de Grises	0.5125	0.7121	0.6345	0.0535
3	Gradiente Morfológico	0.3330	0.5848	0.5256	0.0567
4	Prewitt	0.5827	0.7031	0.6427	0.0288
5	Sobel	0.5246	0.7151	0.6321	0.0483
6	Grad. Morf. Difuso	0.3480	0.6530	0.5815	0.0551
7	Sobel Difuso	0.5727	0.7252	0.6693	0.0343
8	Sobel Color Difuso	0.5196	0.7202	0.6428	0.0524
9	Prewitt Difuso	0.4885	0.7312	0.6473	0.0581

Tabla 5.20. Resultado de entrenamiento CNN-IV aplicado a base de datos CTSD.

No.	Preprocesamiento	Mínimo	Máximo	Promedio	Desviación Estd.
1	Color	0.4794	0.6740	0.5645	0.0521
2	Escala de Grises	0.4824	0.6530	0.5565	0.0532
3	Gradiente Morfológico	0.2207	0.5807	0.5030	0.0578
4	Prewitt	0.5206	0.6229	0.5648	0.0250
5	Sobel	0.2628	0.6560	0.5701	0.0658
6	Grad. Morf. Difuso	0.4885	0.5858	0.5222	0.0207
7	Sobel Difuso	0.5446	0.6379	0.5811	0.0216
8	Sobel Color Difuso	0.5035	0.5988	0.5428	0.0200
9	Prewitt Difuso	0.5577	0.6610	0.5928	0.0291

5.3.3.2. Análisis estadístico para base de datos CTSD

En el caso de esta prueba estadística, los mejores resultados obtenidos fueron aplicando el preprocesamiento de las imágenes a color y Sobel difuso para los primeros tres modelos. En caso de CNN-IV los mejores resultados se obtuvieron aplicando Prewitt difuso y Sobel tradicional.

Tabla 5.21. Análisis estadístico para comparar tipos de preprocesamiento CTSD.

Modelo	Preprocesamiento	$\overline{x_1}$	σ_1	$\overline{x_2}$	σ_2	Valor-Z
CNN-I	Sobel difuso vs. Color	0.7114	0.0263	0.6684	0.0492	**4.2217**
CNN-II	Sobel difuso vs. Color	0.5473	0.0213	0.5318	0.0370	**1.9885**
CNN-III	Sobel difuso vs. Color	0.6693	0.0343	0.6356	0.0565	**2.7926**
CNN-IV	Prewitt difuso vs. Sobel tradicional	0.59284	0.0291	0.5645	0.0521	**2.6011**

5.4. Resultado de convolución con filtros difusos

En esta sección se presentan los resultados al aplicar la experimentación de realizar fusificación en datos y filtros antes de realizar el proceso de convolución. En este caso únicamente se experimentó con las bases de datos BelgiumTS y CTSD. Se utilizaron las funciones de membresía descritas en el capítulo 4, se realizó una combinación entre la utilización de únicamente datos difusos, filtros difusos y utilizar datos y filtros difusos al momento de realizar el entrenamiento. Se realizó el entrenamiento a las cuatro arquitecturas descritas en las Tablas 5.1 a 5.4, haciendo 30 entrenamientos independientes.

Para determinar si existe diferencia significativa entre la utilización de funciones Gaussianas y Triangulares, se desarrolló una prueba estadística con los siguientes parámetros para medir la diferencia de eficacia entre la mejor y la peor función de Membresía utilizada.

Prueba de cola derecha.

- α = 0.05 (Nivel de confidencia del 95%, zona de rechazo z_c =1.96)
- n = 30
- H_0 : La utilización de la función Triangular MF (μ_1) ofrece menor o igual precisión que utilizar Gaussian MF. (μ_2). H_0: $\mu_1 \leq \mu_2$.
- H_a: La utilización de la función Triangular MF (μ_1) ofrece mayor precisión que utilizar Gaussian MF (μ_2).
- H_a: $\mu_1 > \mu_2$(Afirmación).

5.4.1. Aplicación de funciones de membresía a bases de datos BelgiumTS

En esta sección se contiene los resultados la utilización de funciones de membresía aplicados a la base de datos BelgiumTS. En la Tabla 5.22 se muestran los resultados base, donde los modelos fueron entrenados únicamente utilizando las imágenes a escala de grises. Los resultados de aplicar la función Gaussian MF se encuentra plasmado en la Tabla 5.23 a 5.25, los resultados de la utilización de la función de membresía Triangular MF se encuentras en las Tablas 5.26 a 5.28 y por último los resultados de la aplicación de la función de membresía S MF se encuentran en las Tablas 5.29 a 5.31. En la Tabla 5.32 se contiene un resumen de los máximos obtenidos para realizar una comparación entre la información en escala de grises y la información difusa. La Tabla 5.33 contiene los resultados de la prueba estadística al momento de comparar la diferencia entre utilizar función de membresía Gaussian MF y Triangular MF.

Tabla 5.22. Resultados de arquitecturas entrenadas utilizando imágenes en escala de grises aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.9230	0.9746	0.9567	0.0108
CNN-II	0.9135	0.9456	0.9319	0.0081
CNN-III	0.8873	0.9639	0.9396	0.0186
CNN-IV	0.9028	0.9583	0.9341	0.0159

Tabla 5.23. Resultados de arquitecturas entrenadas con datos difusos (Gaussian MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.6750	0.9635	0.9137	0.0554
CNN-II	0.8230	0.9337	0.8931	0.0277
CNN-III	0.7365	0.9520	0.9084	0.0474
CNN-IV	0.8754	0.9520	0.9224	0.0203

Tabla 5.24. Resultados de arquitecturas entrenadas con kernels difusos (Gaussian MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.5159	0.5742	0.5448	0.0148
CNN-II	0.6857	0.7381	0.7154	0.0130
CNN-III	0.5710	0.6151	0.5967	0.0085
CNN-IV	0.6980	0.7520	0.7235	0.0102

Tabla 5.25. Resultados de arquitecturas entrenadas con datos difusos y kernels (Gaussian MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.5218	0.5675	0.5444	0.0127
CNN-II	0.6659	0.7175	0.6949	0.0127
CNN-III	0.5659	0.6095	0.5890	0.0095
CNN-IV	0.6905	0.7341	0.7130	0.0100

Tabla 5.26. Resultados de arquitecturas entrenadas con datos difusos (Triangular MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.9349	0.9750	0.9576	0.0098
CNN-II	0.8734	0.9528	0.9290	0.0156
CNN-III	0.8373	0.9694	0.9396	0.0245
CNN-IV	0.8782	0.9611	0.9353	0.0189

Tabla 5.27. Resultados de arquitecturas entrenadas con kernels difusos (Triangular MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.5472	0.6040	0.5779	0.0129
CNN-II	0.6937	0.7393	0.7209	0.0133
CNN-III	0.5940	0.6317	0.6117	0.0109
CNN-IV	0.7171	0.7504	0.7331	0.0086

Tabla 5.28. Resultados de arquitecturas entrenadas con datos difusos y kernels (Triangular MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.5496	0.6095	0.5830	0.0152
CNN-II	0.7048	0.7492	0.7244	0.0135
CNN-III	0.5698	0.6373	0.6105	0.0158
CNN-IV	0.7060	0.7556	0.7363	0.0120

Tabla 5.29. Resultados de arquitecturas entrenadas con datos difusos (S MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.9349	0.9655	0.9522	0.0079
CNN-II	0.9048	0.9377	0.9279	0.0069
CNN-III	0.8956	0.9587	0.9412	0.0118
CNN-IV	0.9040	0.9500	0.9329	0.0100

Tabla 5.30. Resultados de arquitecturas entrenadas con kernels difusos (S MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.5563	0.6389	0.5907	0.0202
CNN-II	0.7024	0.7583	0.7287	0.0138
CNN-III	0.5893	0.6437	0.6133	0.0133
CNN-IV	0.7238	0.7671	0.7404	0.0108

Tabla 5.31. Resultados de arquitecturas entrenadas con datos difusos y kernels (S MF) aplicado a BelgiumTS.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.5313	0.6373	0.5774	0.0268
CNN-II	0.6873	0.7298	0.7087	0.0110
CNN-III	0.5865	0.6417	0.6090	0.0128
CNN-IV	0.6861	0.7369	0.7103	0.0117

Tabla 5.32. Comparativa entre resultados clásicos y difusos para base de datos BelgiumTS.

Modelo	Escala de Grises	Gaussian MF			Triangular MF			S MF		
		Datos	Kernels	Datos y Kernels	Datos	Kernels	Datos y Kernels	Datos	Kernels	Datos y Kernels
CNN-I	0.9746	0.9635	0.5742	0.5675	**0.9750**	0.6040	0.6095	0.9655	0.6389	0.6373
CNN-II	0.9456	0.9337	0.7381	0.7175	**0.9528**	0.7393	0.7492	0.9377	0.7583	0.7298
CNN-III	0.9639	0.9520	0.6151	0.6095	**0.9694**	0.6317	0.6373	0.9587	0.6437	0.6417
CNN-IV	0.9583	0.9520	0.7520	0.7341	**0.9611**	0.7504	0.7556	0.9500	0.7671	0.7369

Tabla 5.33. Resultado de análisis estadístico comparativo entre funciones de membresía Gaussiana y Triangular para base de datos BelgiumTS.

Modelo	$\overline{x_1}$	σ_1	$\overline{x_2}$	σ_2	Valor-Z
CNN-I	0.9137	0.0554	0.9576	0.0098	**4.2682**
CNN-II	0.8931	0.0277	0.9290	0.0156	**6.1905**
CNN-III	0.9084	0.0474	0.9396	0.0245	**3.2045**
CNN-IV	0.9224	0.0203	0.9353	0.0189	**2.565**

5.4.2. Aplicación de funciones de membresía a bases de datos CTSD

Similares a los resultados de la BelgiumTS los resultados de la experimentación se encuentran divididos conforme a la función de membresía utilizada, la base se encuentra mostrada en la Tabla 5.34, en el caso de la función Gaussian MF se encuentra en las Tablas 5.35 a 5.37, para la

función Triangular MF se encuentran en las Tablas 5.38 a 5.40 y finalmente de aplicar S MF los resultados se plasman en las Tablas 5.41 a 5.43. De igual forma en la experimentación de esta base de datos se tiene la Tabla 5.44 la cual contiene una comparativa de los máximos obtenidos con los distintos métodos de preprocesamiento utilizados. Los resultados de la prueba estadística para determinar la existencia de diferencia significativa al comparar Gaussian MF contra Triangular MF se encuentran en la Tabla 5.45. Al analizar los resultados competentes, de igual forma se compararon los resultados de la utilización de las imágenes en formato RGB contra la fusificación utilizando la función de membresía Triangular solo con los datos, estos resultados se encuentran en la Tabla 5.46

Tabla 5.34. Resultados de arquitecturas entrenadas utilizando imágenes en escala de grises aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.6189	0.7232	0.6674	0.0290
CNN-II	0.4393	0.5707	0.5068	0.0319
CNN-III	0.5125	0.7121	0.6345	0.0535
CNN-IV	0.4824	0.6530	0.5565	0.0532

Tabla 5.35. Resultados de arquitecturas entrenadas con datos difusos (Gaussian MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.4092	0.7232	0.6312	0.0576
CNN-II	0.3049	0.5366	0.4625	0.0528
CNN-III	0.0341	0.6991	0.5534	0.1618
CNN-IV	0.2618	0.6700	0.5508	0.0907

Tabla 5.36. Resultados de arquitecturas entrenadas con kernels difusos (Gaussian MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.2508	0.2949	0.2755	0.0085
CNN-II	0.2718	0.3571	0.3142	0.0186
CNN-III	0.2126	0.2528	0.2274	0.0093
CNN-IV	0.3069	0.3651	0.3386	0.0143

Tabla 5.37. Resultados de arquitecturas entrenadas con datos difusos y kernels (Gaussian MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.2447	0.2849	0.2684	0.0084
CNN-II	0.2598	0.3190	0.2823	0.0126
CNN-III	0.2046	0.2427	0.2253	0.0097
CNN-IV	0.3170	0.3601	0.3358	0.0123

Tabla 5.38. Resultados de arquitecturas entrenadas con datos difusos (Triangular MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.6138	0.7503	0.6930	0.0332
CNN-II	0.4393	0.5928	0.5387	0.0311
CNN-III	0.4895	0.7462	0.6479	0.0595
CNN-IV	0.1354	0.6760	0.5706	0.0939

Tabla 5.39. Resultados de arquitecturas entrenadas con kernels difusos (Triangular MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.2718	0.2979	0.286	0.0069
CNN-II	0.2849	0.3721	0.3166	0.019
CNN-III	0.2156	0.2588	0.2428	0.0096
CNN-IV	0.329	0.3711	0.3501	0.0102

Tabla 5.40. Resultados de arquitecturas entrenadas con datos difusos y kernels (Triangular MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.2748	0.3129	0.2883	0.0086
CNN-II	0.2899	0.337	0.3093	0.0118
CNN-III	0.2217	0.2628	0.2424	0.0101
CNN-IV	0.324	0.3801	0.3518	0.0121

Tabla 5.41. Resultados de arquitecturas entrenadas con datos difusos (S MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.6329	0.7492	0.6935	0.0315
CNN-II	0.4855	0.5848	0.5419	0.0249
CNN-III	0.3400	0.7472	0.6248	0.0779
CNN-IV	0.3771	0.6439	0.5750	0.0488

Tabla 5.42. Resultados de arquitecturas entrenadas con kernels difusos (S MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.2738	0.3069	0.2912	0.0086
CNN-II	0.2688	0.3531	0.3142	0.0172
CNN-III	0.2166	0.2648	0.2443	0.0098
CNN-IV	0.3300	0.3731	0.3540	0.0118

Tabla 5.43. Resultados de arquitecturas entrenadas con datos difusos y kernels (S MF) aplicado a CTSD.

Modelo	Mínimo	Máximo	Promedio	Desv. Est.
CNN-I	0.2728	0.3149	0.2898	0.0095
CNN-II	0.2909	0.3731	0.3099	0.0161
CNN-III	0.2217	0.2618	0.2425	0.0111
CNN-IV	0.3320	0.3791	0.3553	0.0115

Tabla 5.44. Comparativa entre resultados clásicos y difusos para base de datos CTSD.

Modelo	Escala de Grises	Gaussian MF			Triangular MF			S MF		
		Datos	Kernels	Datos y Kernels	Datos	Kernels	Datos y Kernels	Datos	Kernels	Datos y Kernels
CNN-I	0.7232	0.7232	0.2949	0.2849	**0.7503**	0.2979	0.3129	0.7492	0.3069	0.3149
CNN-II	0.5707	0.5366	0.3571	0.3190	**0.5928**	0.3721	0.337	0.5848	0.3531	0.3731
CNN-III	0.7121	0.6991	0.2528	0.2427	**0.7462**	0.2588	0.2628	0.7472	0.2648	0.2618
CNN-IV	0.6530	0.6700	0.3651	0.3601	**0.6760**	0.3711	0.3801	0.6439	0.3731	0.3791

Tabla 5.45. Resultado de análisis estadístico comparativo entre funciones de membresía Gaussiana y Triangular para base de datos CTSD.

Modelo	$\overline{\mathbf{x_1}}$	$\boldsymbol{\sigma_1}$	$\overline{\mathbf{x_2}}$	$\boldsymbol{\sigma_2}$	Valor-Z
CNN-I	0.6312	0.0576	0.6930	0.0332	**5.0875**
CNN-II	0.4625	0.0528	0.5387	0.0311	**6.8203**
CNN-III	0.5534	0.1618	0.6479	0.0595	**3.0018**
CNN-IV	0.5508	0.0907	0.5706	0.0939	0.8293

Tabla 5.46. Resultado de análisis estadístico comparativo entre utilización de imágenes RGB y fusificados con Triangular MF

Modelo	$\overline{\mathbf{x_1}}$	$\boldsymbol{\sigma_1}$	$\overline{\mathbf{x_2}}$	$\boldsymbol{\sigma_2}$	Valor-Z
CNN-I	0.6684	0.0492	0.6930	0.0332	**2.2701**
CNN-II	0.5318	0.0370	0.5387	0.0311	0.7819
CNN-III	0.6356	0.0565	0.6479	0.0595	0.8211
CNN-IV	0.5645	0.0521	0.5706	0.0939	0.3111

5.5. Resultados de experimentación con método de composición difusa

Debido a la complejidad del algoritmo creado para implementar el proceso de composición difusa, en este caso se definieron dos arquitecturas diferentes para realizar pruebas. Las bases de datos utilizadas para experimentación fue un segmento de la ya conocida MNIST [52], donde únicamente se consideraron las categorías cero y uno con 500 elementos para entrenamiento y 200 de prueba. El modelo utilizado para esta base de datos esta descrito por la Tabla 5.47. La segunda base de datos utilizada fue la CTSD utilizando todas sus categorías con un modelo sencillo descrito en la Tabla 5.48.

El experimento realizado para la base de datos MNIST y CTSD se realizó entrenando únicamente por 5 épocas. Para comparar la eficacia de la capa propuesta se comparó contra el mismo modelo utilizando una capa de convolución tradicional. Los resultados obtenidos de la

base de datos MNIST se encuentran descritos en la Tabla 5.49 y los resultados de CTSD en la Tabla 5.50.

Tabla 5.47. Modelo utilizado para base de datos MNIST en composición difusa.

Capa/Tipo	Neuronas/Filtros	Tamaño de filtro
0/Entrada composición/convolución + ReLu	8	3x3
1/Agrupación Máxima	--	2x2
2/Capa densamente conecta	64	--
3/Capa de salida	1	--

Tabla 5.48. Modelo utilizado para base de datos CTSD en composición difusa.

Capa/Tipo	Neuronas/Filtros	Tamaño de filtro
0/Entrada composición/convolución + ReLu	8	3x3
1/Agrupación Máxima	--	2x2
2/Capa densamente conecta	512	--
3/Capa de salida	58	--

Tabla 5.49. Resultados de capa de composición difusa en base de datos MNIST.

Modelo	Capa convolucional	Composición max-min
Precisión	0.5	0.5

Tabla 5.50. Resultados de capa de composición difusa en base de datos CTSD.

Modelo	Convolución	Composición max-min	Composición max-prod
Precisión	0.38816449	0.40220663	0.2929

Capítulo 6. Discusión de Resultados

Este capítulo tiene como objetivo analizar los resultados obtenidos en el contexto de las propuestas planteadas en esta investigación. Se divide en tres subsecciones, en las que se describen los resultados obtenidos en cada una de las propuestas. El subcapítulo 6.1 detalla los resultados obtenidos mediante el preprocesamiento difuso, mientras que el subcapítulo 6.2 presenta los hallazgos con respecto a la implementación de kernels y datos difusos. Finalmente, en el subcapítulo 6.3 se discuten los resultados de la implementación de la composición difusa, con el fin de presentar y analizar los hallazgos de manera detallada.

6.1. Técnicas de preprocesamiento difuso

En los primeros capítulos de este trabajo se optó por trabajar con técnicas de preprocesamiento de imágenes, incluyendo técnicas difusas. Se puede notar en los casos de análisis empleados en las bases de datos GTSRB y BelgiumTS los mejores resultados fueron obtenidos utilizando las imágenes a color y en escala de grises. La Tabla 5.5 contiene los resultados de algunos de los modelos del estado del arte los cuales han trabajado con esta base de datos benchmark. Los resultados obtenidos en la literatura fluctúan entre porcentajes casi perfectos, donde se obtienen valores entre 99.11% hasta 99.71%. En estos casos, la literatura busca la implementación de arquitecturas profundas, tales como VGG-16 con millones de parámetros entrenables, o arquitecturas muy complejas, tal como la propuesta por Garcia y Ciresan, donde las arquitecturas cuentan con capas muy complejas, tales como transformaciones espaciales y contrastes de normalización para aumentar los resultados. En este caso, la investigación giro a obtener resultados competentes, utilizando modelos sustancialmente más pequeños facilitando su ejecución.

Los resultados obtenidos en las Tablas 5.6 a 5.9 para la base de datos GTSRB, muestran que la utilización de técnicas convencionales para la detección de bordes reduce en aproximadamente un 8 % con gradiente morfológico y una reducción del 2 % con Sobel y Prewitt. Del mismo modo, el uso de detectores de bordes difusos reduce la precisión de los modelos de clasificación propuestos en 9% en comparación con las imágenes sin procesar. Los resultados de la prueba estadística realizada están contenidos en la Tabla 5.10, en este análisis se realizó una

comparación entre las imágenes a color y escala de grises, siendo estas dos de las técnicas que presentaron porcentajes de clasificación más altos, por lo cual se realizó la comparativa con ellas. A partir de la representación en escala de grises de las imágenes indican que únicamente en el modelo CNN-II se observa un incremento considerable en la exactitud de la predicción de las señales en el conjunto de prueba, haciendo así que el mejor resultado sea obtenido utilizando las imágenes a color para los modelos CNN-I, CNN-II y CNN-IV.

En la Tabla 5.11 se tienen los resultados del estado del arte, similarmente a la GTSRB, algunos de los modelos utilizados para esa base de datos, tienen resultados favorables, donde los porcentajes de clasificación rondan entre el 98.17% hasta el 99.16%. La experimentación con la base de datos de BelgiumTS que se muestran en las Tablas 5.12 a 5.15 comparten un patrón similar con la base de datos de GTSRB, donde la utilización de técnicas de preprocesamiento, tanto clásicas como difusas reduce la precisión de las predicciones en todas las arquitecturas. Las técnicas tradicionales tales como Sobel o Prewitt reducen la precisión entre un 1 y un 2 % según la arquitectura, en el caso de la detección de bordes por gradiente morfológico tiene un impacto mayor donde se presenta reducción de hasta un 9 % en CNN-III. La utilización de la detección de bordes difusos tiene un impacto negativo en esta base de datos. La consideración de la incertidumbre con técnicas de detección de bordes difusos no proporciona una contribución para mejorar la precisión de las arquitecturas, reduciéndose hasta un 12% con CNN-III. Los resultados del análisis estadístico en la Tabla 5.16 demuestran que no hay evidencia significativa para validar el uso de una técnica de preprocesamiento con ninguno de los modelos.

La base de datos CTSD se diferencia contra GTSRB y BelgiumTS, debido a que esta tiene la particularidad en la cual existe una gran presencia de ruido y elementos externos en el objeto de interés. Dado a esto, los resultados de la base de datos CTSD tienen comportamiento peculiar comparado con las bases de datos posteriores. En este caso, observamos una mejora en la precisión de la predicción de los cuatro modelos con la mayoría de las técnicas de detección de bordes, tanto tradicionales como difusas. El mejor resultado se obtiene utilizando la combinación del modelo CNN-I junto con el detector de borde difuso por Sobel, donde se logra una mejora superior al 3%, con una precisión máxima dando un 76% de predicciones correctas en el conjunto de prueba. Al realizar el análisis de la prueba estadística, se observa que en las cuatro arquitecturas hay evidencia significativa de una mejora en la precisión cuando se utiliza un enfoque de preprocesamiento; se logra mejores resultados con la implementación difusa de Sobel

en las arquitecturas de CNN-I a CNN-III, y con la implementación difuso de Prewitt en CNN-IV. En este caso en particular se observa que los detectores de bordes difusos tienen gran utilidad al emplearse en datos con grandes cantidades de incertidumbre.

6.2. Implementación de convolución con kernels y datos difusos

El segundo acercamiento donde se plantea la utilización de datos y filtros difusos se obtienen resultados diversos comparado con la información clásica (escala de grises). En este caso solo se experimentó con las bases de datos BelgiumTS y CTSD debido a la cantidad de elementos que contiene.

Los resultados obtenidos por la base de datos BelgiumTS en las Tablas 5.22 a 5.31, muestran particularmente una pérdida de precisión al momento de utilizar funciones de membresía Gaussiana (Tablas 5.23 a 5.25) y S (Tablas 5.29 a 5.31), esto cuando son aplicadas a los datos, los filtros y en su combinación, de igual forma con la función de membresía Triangular no muestra mejoras al utilizar filtros o la combinatoria de datos y filtros difusos (Tablas 5.27 y 5.28), esto cuando es comparado con los resultados obtenidos al realizar la clasificación de los datos en escala de grises mostrados en la Tabla 5.22. En cambio, al utilizar únicamente datos difusos con la función de membresía Triangular (Tabla 5.26) se aprecia que la precisión mejora significativamente al comparar con la utilización de datos crudos (escala de grises) de los cuatro modelos propuestos. La Tabla 5.32 muestra como la utilización de datos fusificados es la mejor alternativa al realizar esta experimentación, dado que a pesar de la pérdida de precisión al utilizar Gaussian MF y S MF, la disminución de precisión es mínima. En el caso de la utilización de Triangular MF con datos difusos se observa que las cuatro arquitecturas propuestas incrementan el porcentaje de clasificación.

El análisis estadístico realizado en la Tabla 5.33 muestra que la comparación entre la utilización de la función de membresía Gaussiana y Triangular no genera evidencia significativa para determinar que alguna de ellas es mejor al momento de realizar la tarea de clasificación.

El uso de técnicas de datos y filtros difusos en el caso de estudio de la base de datos CTSD muestra un patrón similar al de BelgiumTS según los resultados registrados en las Tablas 5.34 a 5.43. La aplicación de funciones de membresía en datos, kernels y su combinación con funciones Gaussian MF (Tablas 5.35 a 5.37) y S MF (Tablas 5.41 a 5.43) y la utilización de Triangular MF

con kernels y combinación de kernels y datos difusos (Tablas 5.39 a 5.40), resulta en una disminución en la precisión máxima. Sin embargo, al aplicar la Triangular MF a los datos de entrada del modelo, similar a la BelgiumTS, se logra un aumento en la precisión máxima al clasificar hasta un 3% en la CNN-III. La Tabla 5.44, que contiene el resumen de los máximos obtenidos en la experimentación, demuestra que el uso de datos fusificados es la mejor opción para incluir información difusa en el modelo, especialmente con la función Triangular que presenta una mejora constante en los resultados en comparación con Gaussian MF y S MF, que presentan una pérdida no significativa. Los resultados de la prueba estadística mostrados en la Tabla 5.45 indican que se logra una mejora significativa al comparar la utilización de la función de membresía Triangular contra la Gaussiana en algunos de los modelos propuestos.

Dados los resultados obtenidos, se decidió realizar un segundo análisis estadístico, donde se comparan los resultados de utilizar las imágenes en formato RGB mostradas en las Tablas 5.17 a 5.20 contra los datos en espectro difuso por Triangular MF, esto debido a que las precisiones máximas y medias superan a los datos a color en todos los casos. Los resultados de la Tabla 5.46 se muestra que únicamente con el modelo CNN-I se encuentra una diferencia significativa al utilizar este tipo de tratamiento difuso a los datos, haciéndolo una alternativa optima a no utilizar ningún tipo de procesamiento.

6.3. Implementación de composición

En esta implementación se utilizó la propuesta de una nueva capa de filtrado para realizar extracción de características, buscando sustituir la capa convencional de convolución en una CNN. Debido a la complejidad del algoritmo propuesto, donde se realiza la operación de filtrado de ventana sobre volumen, similar a la capa de convolución, pero con la sustitución del producto escalar por la operación de composición, se obtuvieron tiempos de entrenamiento inmensos ya que no existe una implementación optimizada en GPU paralelizando el proceso. Debido a esto se optó por reducir los modelos y realizar experimentación sobre dos bases de datos, la modificación de MNIST y CTSD, la última ya que es la más pequeña y compleja en el enfoque de clasificación de señales de tránsito.

El caso de estudio aplicado con la base de datos MNIST los cuales se muestran en la Tabla 5.49 podemos observar que se logra obtener resultados similares utilizando la capa tradicional de

convolución, así como la propuesta de composición. Obteniendo un 50% de precisión en ambos modelos. Cabe destacar que al ser una muestra de únicamente dos clases de los modelos convergen casi inmediatamente al iniciar el entrenamiento.

En el caso de la CTSD mostrados en la Tabla 4.50 podemos observar que se obtienen resultados distintos, donde la aplicación de la capa propuesta de composición alcanza mejores resultados que la capa tradicional de convolución, utilizando la misma estructura y cantidad de filtros, únicamente cambiando la operación realizada. En este caso se obtiene un 1.4% de mejora en los datos de prueba con el experimento realizado. La pérdida final obtenida de la capa tradicional es de 9.4183, mientras que la pérdida obtenida con el modelo propuesto es de 2.7326, a lo largo del entrenamiento se visualizó un comportamiento más estable con la capa propuesta.

Capítulo 7. Conclusiones y Trabajo Futuro

Los casos de estudio aplicados sobre las bases de datos GTSRB y BELGIUMTS demostraron que las etapas de preprocesamiento no tienen gran impacto en la precisión del modelo debido a que este porcentaje no mejora significativamente.

A diferencia la base de datos CTSD muestra una mejora significativa aplicando técnicas de preprocesamiento, especialmente el preprocesamiento de bordes difusos por Sobel mejorando dos modelos propuestos.

Se plantea que la utilización de detectores de bordes reduce significativamente la información de la imagen causando que se pierdan características esenciales de las señales de tránsito, haciendo que en algunos de estudio la precisión baje drásticamente.

La capa de expulsión, que en algunos casos es muy agresivas, ayuda a combatir problemas de sobre entrenamiento haciendo que los modelos más complejos logren generalizar mejor contra la información de prueba.

La utilización de funciones de membresía ofrece una mejora en la precisión de los modelos, especialmente la función Triangular, la cual benefició particularmente a la base de datos CTSD mejorando significativamente sus resultados. El uso de información difusa en kernels, así como en kernels y datos tiene un impacto negativo al realizar el entrenamiento de los modelos.

La utilización de la capa de composición difusa no demostró una mejora significativa al momento de ser aplicado al caso de estudio reducido de MNIST. En el caso de la aplicación a la CTSD se logró obtener una mejora del 1.4% en el experimento realizado, mostrando una ventaja en bases de datos multiclase con datos más complejos. Cabe destacar que esta implementación, dada su complejidad se maneja como prueba de concepto, donde se buscaba obtener un resultado parcial para seguir con el desarrollo de esta capa propuesta, en este caso mostrando potencial para su estudio.

Como trabajo futuro se planea mejorar la implementación realizada la capa de composición para aumentar su rendimiento haciendo uso ya sea de paralelización utilizando multiprocesadores o una aplicación definida en CUDA para la utilización de GPU’s en alguna librería popular de aprendizaje profundo. Esto con el fin de poder realizar una experimentación más amplia, aplicando la capa propuesta a otras bases de datos o enfoques, como lo es el procesamiento de texto o audio.

Capítulo 8. Referencias

[1] J. Rawat, A. Singh, H. S. Bhadauria, J. Virmani and J. S. Devgun, "An Investigative Study on Early Diagnosis of Prostate Cancer Using Neuro-Fuzzy Classification System for Pattern Recognition," *Arab J Sci Eng,* no. 43, p. 7041–7058, 2018.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Transactions on Intelligent Vehicles ,* vol. 1, no. 1, pp. 33 - 55, 2016 .

[3] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Network," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li and K. F.-F. L. Li, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, pp. 248 - 255, 2009.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[6] K. P. Korshunova, "A Convolutional Fuzzy Neural Network for Image Classification," in *2018 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)*, Vladivostok, Russia, 2018.

[7] M.-J. Hsu, Y.-H. Chien, W.-Y. Wang and H. Chen-Chien, "A Convolutional Fuzzy Neural Network Architecture for Object Classification with Small Training Database," *International Journal of Fuzzy Systems,* vol. 22, pp. 1-10, 2020.

[8] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision,* vol. 88, pp. 303-308, 2009.

[10] D. E. Diamantis and D. K. Iakovidis, "Fuzzy Pooling," *IEEE Transactions on Fuzzy Systems,* vol. 29, no. 11, pp. 3481 - 3488, 2020.

[11] T.-L. Nguyen, S. Kavuri and M. Lee, "A fuzzy convolutional neural network for text sentiment analysis," *Special Section: Green and Human Information Technology,* vol. 35, no. 6, pp. 6025-6034,, 2018.

[12] D. Sundararajan, Digital Image Processing A Signal Processing and Algorithmic Approach, Montereal: Springer, 2017.

[13] R. Szeliski, Image Processing, Switzerland: Springer, 2022.

[14] Navdeep, V. Singh, A. Rani and S. Goyal, "An improved hyper smoothing function based edge detection algorithm for noisy images," *Journal of Intelligent & Fuzzy Systems,,* vol. 38, no. 5, pp. 6325-3665, 2020.

[15] H. Singh and T. Kaur, "Novel Method for Edge Detection for Gray Scale Images using VC++," *International Journal of Advanced Computer Research,* vol. 47, pp. 1-6, 2013.

[16] V. Maksimović, B. Jakšić, M. Petrović, P. Spalević and S. Panić, "New approach to edge detection on different level of wavelet decomposition," *Computing and Informatics,* vol. 39, pp. 1067-1090., 2019.

[17] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 8, p. 679–698, 1986.

[18] R. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research,* vol. 4, p. 315–328, 1971.

[19] I. Sobel, *Camera Models and Perception,* Stanford: Stanford University, 1970.

[20] J. M. S. Prewitt, "Object enhancement and extraction. Picture Analysis and Psychopictorics," *Academic Press,* vol. 1, p. 75–149., 1970.

[21] P. Melin, C. Gonzalez, J. Castro, O. Mendoza and O. Castillo, "Edge Detection Method for Image Processing Based on Gen-eralized Type-2 Fuzzy Logic," *IEEE Trans. Fuzzy Syst,* vol. 22, pp. 1515-1525, 2014.

[22] P. Melin, O. Mendoza and O. Castillo, "An improved method for edge detection based on interval type-2 fuzzy logic," *Expert Systems with Applications,* vol. 37, p. 8527– 8535, 2010.

[23] G. Martínez, C. Gonzalez, O. Mendoza and P. Melin, "General Type-2 Fuzzy Sugeno Integral for Edge Detection," *Journal of Imaging,* vol. 5, pp. 1-20, 2019.

[24] S. A., D. J. Shiny, P. R. and S. Dioline, "Performance evaluation of adaptive neuro fuzzy system (ANFIS) over fuzzy inference system (FIS) with optimization algorithm in de-noising of images from salt and pepper noise," *Journal of Ambient Intelligence and Humanized Computing,* 2021.

[25] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics,* vol. 5, pp. 115-133, 1943.

[26] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review,* vol. 65, no. 6, pp. 386-408, 1958.

[27] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *Nature,* vol. 521, pp. 436-444, 2015.

[28] B. Ding, H. Qian and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *2018 Chinese Control And Decision Conference (CCDC)*, Shenyang, 2018, pp. 1836-1841.

[29] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

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

[31] Q. Wang, Y. Ma, K. Zhao and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine," *Annals of Data Science,* vol. 9, pp. 187-212, 2020.

[32] S. Skansi, "Convolutional Neural Networks," in *ntroduction to Deep Learning From Logical Calculus to Artificial Intelligence*, Zagreb, Springer, 2018, pp. 121-133.

[33] L. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human

reasoning and fuzzy logic," *Fuzzy Sets and Systems,* vol. 90, no. 2, pp. 111-127, 1997.

[34] J.-S. R. Jang, C. T. Sun and E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Pearson, 1997.

[35] E. Mamdani, "Analysis of fuzzy logic controller," *Fuzzy sets and Systems,* vol. 1, no. 1, pp. 29-44, 1978.

[36] R. Siregar, M. Zarlis and Z. Situmorang, "A comparison tsukamoto and mamdani methods in fuzzy inference system for determining nutritional toddlers," in *The 6th International Conference on Cyber and IT Service Management (CITSM 2018)*, Medan, 2018.

[37] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 15, no. 1, pp. 116-132, 1985.

[38] E. Pintelas, M. Liaskos, E. L. Ioannis, S. Kotsiantis and P. Pintelas, "A Novel explainable image classification framework: case study on skin cancer and plant disease prediction," *Neural Computing and Applications,* no. 33, pp. 15171-15189, 2021.

[39] H. Hangras, "Toward Human-Understandable, Ai," *Computer,* vol. 51, no. 9, pp. 28-36, 2018.

[40] J. Amann, A. Blasimme, E. Vayena, D. Frey and V. I. Madai, "Explainability for artificial intelligence in healthcare a multidisciplinary perspective," *BMC Medical Informatics and Decision Making,* vol. 20, 2020.

[41] A. B. Arrietaa, N. Díaz-Rodríguez, J. Del Ser, A. Benneto, S. Tabikg, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatilaf and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies,," *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,* vol. 58, pp. 82-115, 220.

[42] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *Proceedings of the IEEE International Joint Conference on Neural Network*, San Jose, 1453–1460.

[43] R. Timofte, K. Zimmermann and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," *Machine Vision and Applications,* no. 25, pp. 633-647, 2011.

[44] Y. Zhang, Z. Wang, Y. Qi, J. Liu and J. Yang, "CTSD: A Dataset for Traffic Sign Recognition in Complex Real-World Images," in *2018 IEEE Visual Communications and Image Processing (VCIP)*, Taichung, 2018.

[45] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis and J. Dean, "TensorFlow: A System for Large-Scale," in *12th USENIX Symposium on Operating Systems Design*, Savannah, 2016.

[46] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools,* 2000.

[47] Y. Yu, J. Li, C. Wen, H. Guan, H. Luo and C. Wang, "Bag-of-visual-phrases and hierarchical deep models for traffic sign detection and recognition in mobile laser scanning data," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 113, pp. 106-123, 2016.

[48] J. Jin, K. Fu and C. Zhang, "Traffic Sign Recognition With Hinge Loss Trained Convolutional Neural Networks," *IEEE Transactions on Intelligent Transportation Systems,* vol. 15, no. 5,

pp. 1991-2000, 2014.

[49] F. Jurišić, I. Filković and Z. Kalafatić, "Multiple-dataset traffic sign classification with OneCNN," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, Malaysia, 2015.

[50] Á. Arcos-García, J. A. Álvarez-García and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," *Neural Networks,* vol. 99, pp. 158-165, 2018.

[51] A. Jain, A. Mishra, A. Shukla y R. Tiwari, «A Novel Genetically Optimized Convolutional Neural Network for Traffic Sign Recognition: A New Benchmark on Belgium and Chinese Traffic Sign Datasets,» *Neural Processing Letters,* vol. 50, p. 3019–3043, 2019.

[52] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[53] S. Houben, J. Stallkamp, J. Salmen, J. Salmen, S. M and C. Igel, "Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, 2013.

Capítulo 9. Anexos

Anexo 1. Detección de bordes por gradiente morfológico difuso.

```
function fis=deltaedgesit1(clow,cmedium,chigh,cfondo,cborde)

NumMf = [3 3 3 3 3];
MfType = str2mat('gaussmf','gaussmf','gaussmf','gaussmf',...
                 'gaussmf');

sd=chigh./4;
sefondo=abs(cfondo-cborde)/2;
seborde=abs(cfondo-cborde)/2;

if(length(size(clow))==1)
MfParams{1}=[20 clow;20 cmedium;20 chigh];
MfParams{2}=[20 clow;20 cmedium;20 chigh];
MfParams{3}=[20 clow;20 cmedium;20 chigh];
MfParams{4}=[20 clow;20 cmedium;20 chigh];
else
MfParams{1}=[sd(1) clow(1) ;sd(1) cmedium(1); sd(1) chigh(1)];
MfParams{2}=[sd(2) clow(2) ;sd(2) cmedium(2); sd(2) chigh(2)];
MfParams{3}=[sd(3) clow(3) ;sd(3) cmedium(3); sd(3) chigh(3)];
MfParams{4}=[sd(4) clow(4) ;sd(4) cmedium(4); sd(4) chigh(4)];
end
MfParams{5}=[sefondo cfondo;seborde cborde];

MfParams{1};
MfParams{2};
MfParams{3};
MfParams{4};
```

```
MfParams{5};

fis = newfis('edges','mamdani');

fis = addvar(fis,'input','G1',[-100 300]);
fis = addvar(fis,'input','G2',[-100 300]);
fis = addvar(fis,'input','G3',[-100 300]);
fis = addvar(fis,'input','G4',[-100 300]);
fis = addvar(fis,'output','Edges',[min([cfondo cborde]) max([cfondo
cborde])]);
%
fis = addmf(fis,'input',1,'LowG1',MfType(1,:),MfParams{1}(1,:));
fis = addmf(fis,'input',1,'MiddleG1',MfType(1,:),MfParams{1}(2,:));
fis = addmf(fis,'input',1,'HighG1',MfType(1,:),MfParams{1}(3,:));
%
fis = addmf(fis,'input',2,'LowG2',MfType(2,:),MfParams{2}(1,:));
fis = addmf(fis,'input',2,'MiddleG2',MfType(2,:),MfParams{2}(2,:));
fis = addmf(fis,'input',2,'HighG2',MfType(2,:),MfParams{2}(3,:));
%
fis = addmf(fis,'input',3,'LowG3',MfType(3,:),MfParams{3}(1,:));
fis = addmf(fis,'input',3,'MiddleG3',MfType(3,:),MfParams{3}(2,:));
fis = addmf(fis,'input',3,'HighG3',MfType(3,:),MfParams{3}(3,:));

fis = addmf(fis,'input',4,'LowG4',MfType(4,:),MfParams{4}(1,:));
fis = addmf(fis,'input',4,'MiddleG4',MfType(4,:),MfParams{4}(2,:));
fis = addmf(fis,'input',4,'HighG4',MfType(4,:),MfParams{4}(3,:));

fis = addmf(fis,'output',1,'Background',MfType(5,:),MfParams{5}(1,:));
fis = addmf(fis,'output',1,'Edge',MfType(5,:),MfParams{5}(2,:));
```

```
ruleList= [ 3   3   3   3   2  1   2;
            2   2   2   2   2  1   2;
            1   1   1   1   1  1   1
    ];

fis=addrule(fis,ruleList);

showrule(fis);
end



function [Imatriz,dim]=lee_imagen(archivo)
    Imatriz=imread(archivo);
    dim=size(Imatriz);
    if length(dim)>2
       if dim(3)==3
        Imatriz=rgb2gray(Imatriz);
       end
    end
end

% OPERADOR GRADIENTE + LOGICA DIFUSA TIPO 1
clear all;
warning('off')
input_dir = '/home/cesariux2596/Dev/Datasets/CGD/CGD_no_split/';
output_dir = '/home/cesariux2596/Dev/Datasets/CGD/CGD_fuzzy_sobel/';
tic
for folder = 0:5
    source_folder = strcat(input_dir, string(folder));
    source_folder = strcat(source_folder, '/');
    output_folder = strcat(output_dir, string(folder));
```

```
    output_folder = strcat(output_folder, '/');
    srcFiles = strcat(source_folder, '*.png');
    srcFiles = dir(srcFiles);

    for i = 1:length(srcFiles)
        filename = strcat(source_folder, srcFiles(i).name);

        Imatriz = lee_imagen(filename); %lee la imagen y la convierte
a tonos de gris si es necesario
        cfondo = -5;
        cborde = 4.5;
        K = imresize(Imatriz, [152 152]);

        % Detecta bordes con el metodo de Sobel tradicional
        [D1, D2, D3, D4] = absdelta(K);
        BordesDM = D1 + D2 + D3 + D4;

        % Detecta bordes con el metodo de Sobel+FIS1
        [ren, col] = size(D1);
        vd1 = reshape(D1, 1, ren * col);
        vd2 = reshape(D2, 1, ren * col);
        vd3 = reshape(D3, 1, ren * col);
        vd4 = reshape(D4, 1, ren * col);

        clow = [min(vd1), min(vd2), min(vd3), min(vd4)];
        chigh = [max(vd1), max(vd2), max(vd3), max(vd4)];
        cmedium = [(clow + chigh) ./ 2];

        % Crea y evalua el fist1
        fis = deltaedgesit1(clow, cmedium, chigh, cfondo, cborde);
        vector_FIS = evalfis([vd1; vd2; vd3; vd4]', fis);
```

```
        BordesFIS = reshape(vector_FIS, ren, col);

    end

end
toc
    end

end
toc
```

Anexo 2. Fusificación de datos.

```
import skfuzzy as fuzz
import pandas as pd
import numpy as np

input_universe_img = np.arange(0,256,1)
mf_img = fuzz.trimf(input_universe_img,[0,255,255])
def fuzz_data(img):
    out = np.zeros_like(img)
    for y in range(img.shape[0]):
        for x in range(img.shape[1]):
            for z in range(img.shape[2]):
                out[y][x][z] =
fuzz.interp_membership(input_universe_img, mf_img, img[y][x][z])
    return out

kernel_input_universe = np.arange(-1,1,0.0001)
kernel_input_int = fuzz.trimf(kernel_input_universe,[-1,1,1])
def fuzz_window(kernel):
```

```
    out = np.zeros_like(K.get_value(kernel))
    for f in range(kernel.shape[0]):
        for y in range(kernel.shape[1]):
            for x in range(kernel.shape[2]):
                out[f][y][x] =
fuzz.interp_membership(kernel_input_universe, kernel_input_int,
kernel[f][y][x])
    return out
```

Anexo 3. Composición Min-Max a N dimensiones

```
from keras import layers, models
import tensorflow as tf
import keras.backend as K
from tensorflow import keras
import keras.layers as layers
import numpy as np


@tf.function
def nd_max_comp(img, ker):
    img = tf.cast(img, tf.float32)
    ker = tf.cast(ker, tf.float32)

    y = tf.shape(img)[0]  # Y size from image
    x = tf.shape(ker)[1]  # X size from Kernel
    in_dim = tf.shape(ker)[2]  # Kernel Input dims (matches image
dims)
    out_dim = tf.shape(ker)[3]  # Kernel output dims

    out = tf.zeros([y,x,out_dim],tf.float32)
```

```
    for dim in tf.range(in_dim):
        for o_dim in tf.range(out_dim):
            for xx in tf.range(x):
                for yy in tf.range(y):
                    out =
out,[[xx,yy,o_dim]],tf.reshape(tf.reduce_min(tf.maximum(img[yy,:,dim],
tf.transpose(ker)[o_dim,dim,xx])),[1,]))

    return tf.transpose(out)
```

Anexo 4. Composición Max-Product a N dimensiones

```
from keras import layers, models
import tensorflow as tf
import keras.backend as K
from tensorflow import keras
import keras.layers as layers
import numpy as np

@tf.function
def nd_max_comp(img, ker):
    img = tf.cast(img, tf.float32)
    ker = tf.cast(ker, tf.float32)

    y = tf.shape(img)[0]  # Y size from image
    x = tf.shape(ker)[1]  # X size from Kernel
    in_dim = tf.shape(ker)[2]  # Kernel Input dims (matches image
dims)
    out_dim = tf.shape(ker)[3]  # Kernel output dims
```

```
    out = tf.zeros([y,x,out_dim],tf.float32)
    for dim in tf.range(in_dim):
        for o_dim in tf.range(out_dim):
            for xx in tf.range(x):
                for yy in tf.range(y):
                    out =
tf.update(out,[[xx,yy,o_dim]],tf.reshape(tf.reduce_max(tf.math.multipl
y(img[yy,:,dim], ker[o_dim,dim,xx]))))
    return tf.transpose(out)
```

Anexo 5. Capa de composición propuesta.

```
from keras import layers, models
import tensorflow as tf
import keras.backend as K
from tensorflow import keras
import keras.layers as layers
import numpy as np

@tf.function
def comp_layer(pre_layer, kernel, strides=[1,1]):

    kernel_h = tf.shape(kernel)[0]
    kernel_w = tf.shape(kernel)[1]
    kernel_out_ch = tf.shape(kernel)[3]
    batch = tf.shape(pre_layer)[0]
    input_h = tf.shape(pre_layer)[1]
    input_w = tf.shape(pre_layer)[2]
    out_h = input_h-kernel_h
    out_w = input_w-kernel_w
    out_h = input_h-kernel_h
    out_w = input_w-kernel_w
```

```
    conv_result = tf.zeros([batch,out_h, out_w, kernel_out_ch],
tf.float32)
    for b in tf.range(batch):
        for i in tf.range(out_h):
            for j in tf.range(out_w):
                gc.collect()
                conv_result = tf.update(conv_result,[[[b,i,j]]],

[[tf.reduce_max(nd_max_comp(pre_layer[b,

i:i+3,j:j+3,:],kernel))]])
    return conv_result

class FuzzyConv2D(tf.keras.layers.Layer):
    def __init__(self, filters, kernel_size, activation):
        super(FuzzyConv2D, self).__init__()
        self.filters = filters
        self.kernel_size = kernel_size
        self.activation = activation
        self.kernel_initializer=tf.initializers.GlorotUniform()
        self.bias_initializer=tf.initializers.Zeros()

    def get_config(self):
        config = super().get_config().copy()
        config.update({
            'filters': self.filters,
            'kernel_size': self.kernel_size,
            'activation': self.activation
        })
        return config
```

```
    def build(self, input_shape):
        *_, n_channels = input_shape
        self.kernel = tf.Variable(

initial_value=self.kernel_initializer(shape=(*self.kernel_size,
                                                    n_channels,

self.filters),

dtype='float32'), trainable=True)

    def call(self, layer_input):

        gc.collect()
        return comp_layer(layer_input,self.kernel)
```