industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Sistema de percepción de elementos viarios usando técnicas de visión por computador para aplicación en conducción autónoma

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INDUSTRIAL

Autor: **Antonio Ros García**
Director: Pedro Javier Navarro Lorente
Codirector: Carlos Fernández Andrés

Cartagena, Diciembre de 2019

Resumen

El objetivo de este trabajo es diseñar, implementar y evaluar diferentes sistemas de detección de objetos para sistemas de conducción autónoma, empleando únicamente la información obtenida por una cámara. Para ello se han estudiado diferentes algoritmos de visión artificial y de clasificación como las redes neuronales artificiales, a partir de los cuales se han creado sistemas de detección de líneas viales y señales de tráfico. Además, se ha desarrollado una aplicación en Python que permitirá obtener una imagen del entorno y detectar señales de tráfico.

Abstract

The objective of this work is to design, implement and evaluate different object detection systems for autonomous driving systems, using the information obtained by a camera. For this, different artificial vision and classification algorithms such as artificial neural networks have been studied, from which road line and traffic signal detection systems and have been developed. In addition, a signal detection system will be implemented that is capable of obtaining information from the environment using a camera and searching for traffic signals contained in the images obtained.

Agradecimientos

Quisiera agradecer a varias personas la ayuda que me han prestado en la realización de este Trabajo Fin de Máster.

En primer lugar, a mi director, Pedro Javier Navarro por su inestimable ayuda y enseñanzas, sin las cuales hubiera sido muy difícil la realización de este proyecto. A mis padres, pareja y amigos, por todo su apoyo, por sus sabios consejos y su comprensión. Siempre habéis estado ahí para ayudarme y sin ellos nada de esto hubiera sido posible.

Muchas gracias a todos.

Antonio Ros García.

Índice de contenido

CAPÍTULO 1. INTRODUCCIÓN Y OBJETIVOS 15

1.1 Introducción 16

1.2 Objetivos del proyecto 16

1.3 Objetivos 18

1.4 Descripción de los capítulos 18

CAPÍTULO 2. ESTADO DEL ARTE 21

2.1 Introducción 22

2.2 Fundamentos 22

2.2.1 Formación y representación de la imagen 22

2.3 Etapas en un proceso de visión artificial 23

2.4 Espacio de color 24

2.5 Detección de bordes 26

2.5.1 Algoritmo de Canny 26

2.5.1.1 Obtención del gradiente 26

2.5.1.2 Supresión no máxima al resultado del gradiente 26

2.5.1.3 Histéresis de umbral a la supresión no máxima 28

2.5.2 Operador Sobel 28

2.6 Transformada de Hough 30

2.7 Conceptos de inteligencia artificial 31

2.7.1 Aprendizaje automático 31

2.7.2 Aprendizaje profundo 32

2.8 Redes neuronales 33

2.8.1 Redes Neuronales Artificiales 34

2.8.1.1 Redes de capa simple 35

2.8.1.1 Redes multicapa 35

2.8.1.2 Conceptos 36

2.8.1.3 Aplicaciones 37

CAPÍTULO 3. SISTEMA DE DETECCIÓN DE LÍNEAS VIALES 39

3.1 Introducción 40

3.2 Sistemas de detección de líneas viales 40

3.2.1 Detección de líneas rectas 40

3.2.1.1 Análisis de la imagen 41

3.2.1.2 Cambio de espacio de color 42

3.2.1.3 Selección de la región de interés 42

3.2.1.4 Algoritmo de Canny 43

3.2.1.5 Transformada de Hough 43
3.2.1.6 Finalmente 44
3.2.2 Detección de líneas curvas 44
3.2.2.1 Corrección de distorsión 45
3.2.2.2 Cambio de perspectiva 45
3.2.2.3 Operador Sobel 46
3.2.2.4 Detección de pico de histograma y ventana deslizante 46
3.2.2.5 Ajuste de curvas 46
3.2.3 Sistemas de detección de líneas mediante aprendizaje profundo 47
CAPÍTULO 4. SISTEMA DE DETECCIÓN DE SEÑALES DE TRÁFICO 49
4.1 Introducción 50
4.2 Obtención de datos 50
4.3 Extracción de los datos 51
4.4 Preprocesamiento de los datos 54
4.5 Redes neuronales convolucionales 55
4.6 Arquitecturas de las CNN 60
4.6.1 LeNet 62
4.6.2 VGGNet 62
4.7 Modelo de entrenamiento y evaluación 63
4.8 Pruebas del modelo 66
4.8.1 Uso en ejemplos de prueba 66
4.8.2 Uso en nuevas imágenes 67
CAPÍTULO 5. IMPLEMENTACIÓN DEL SISTEMA DE DETECCIÓN DE SEÑALES EN UN SISTEMA AUTÓNOMO 71
5.1 Introducción 72
5.2 Elección de cámara 72
5.3 Algoritmos de imágenes piramidales y ventana deslizante 73
5.4 Implementación de un sistema de detección de señales 74
5.5 Pruebas del algoritmo 77
CAPÍTULO 6. CONCLUSIÓN Y TRABAJOS FUTUROS 81
6.1 Introducción 82
6.2 Conclusión 82
6.3 Trabajos futuros 83
Bibliografía 85

Índice de figuras

Figura 1.1 El ojo de una maquina 17
Figura 1.2 Niveles de conducción autónoma 17
Figura 2.1 Diagrama de bloques de las etapas de un sistema de visión artificial 24
Figura 2.2 Pixeles de una pantalla 24
Figura 2.3 Cono del modelo HSV 25
Figura 2.4 Máscaras de convolución para el obtener el filtro gaussiano 26
Figura 2.5 Gradiente en dirección horizontal y vertical de una imagen 27
Figura 2.6 Resultado de la detección de bordes mediante el algoritmo de Canny 28
Figura 2.7 Uso operador Sobel 29
Figura 2.8 (a) Recta en el plano xy; (b) Espacio de parámetros. 30
Figura 2.9 (a) Parametrización de las líneas en el plano xy; (b) Curvas sinusoidales en el plano ρ θ 31
Figura 2.10 Etapas de la inteligencia Artificial 31
Figura 2.11 Estructura de una neurona 33
Figura 2.12 Esquema de una red neuronal 34
Figura 2.13 Esquema de una Red Neuronal Artificial de capa simple 35
Figura 2.14 Esquema de una Red Neuronal Artificial multicapa 36
Figura 3.1 Autovía Cartagena-Murcia 41
Figura 3.2 Flujograma del sistema de detección de líneas viales 41
Figura 3.3 Cambio a espacio de color HSV 42
Figura 3.4 Selección de la región de interés 43
Figura 3.5 Detección de bordes algoritmo de Canny 43
Figura 3.6 Líneas de carril 44
Figura 3.7 Detección de las líneas viales 44
Figura 3.8 Flujograma de detección de líneas curvas 45
Figura 3.9 Cambio de perspectiva 46
Figura 3.10 Ejemplo histograma de imagen binaria de líneas viales 46
Figura 3.11 Detección de líneas curvas 47
Figura 3.12 Red neuronal propuesta en “Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks” 48
Figura 4.1 Ejemplo de imágenes de señales del dataset 50
Figura 4.2 Ejemplos de imágenes de entrenamiento 51
Figura 4.3 Histograma del número de ejemplos de entrenamiento 52
Figura 4.4 Histograma del número de ejemplos de validación 52
Figura 4.5 Histograma del número de ejemplos de prueba 52
Figura 4.6 Imágenes en escala de grises del dataset 54

Figura 4.7 Imágenes en las que se ha aplicado la ecualización del histograma local del dataset 54

Figura 4.8 Red Neuronal Convolucional 55

Figura 4.9 Izquierda: Ejemplo de una entrada de una imagen de 32x32x3 pixeles con través de 5 neuronas conectadas. Derecha: Cálculos que se producen dentro de la neurona para obtener los pesos 56

Figura 4.10 Funciones de activación 57

Figura 4.11 Izquierda: La capa de agrupación reduce el volumen espacialmente, independientemente en cada segmento de profundidad del volumen de entrada. Izquierda: en este ejemplo, el volumen de entrada de tamaño [224x224x64] se agrupa con el tamaño de filtro 2, paso 2 en el volumen de salida de tamaño [112x112x64]. Tenga en cuenta que la profundidad del volumen se conserva. Derecha: La operación de disminución de resolución más común es max, dando lugar a la agrupación máxima, que se muestra aquí con un paso de 2. Es decir, cada máximo se toma sobre 4 números (pequeño cuadrado de 2x2). 58

Figura 4.12 Proceso de clasificación softmax 59

Figura 4.13 Estructura LeNet 60

Figura 4.14 Estructura AlexNet 60

Figura 4.15 Estructura VGGNet 61

Figura 4.16 Estructura LeNet para detectar señales de tráfico 62

Figura 4.17 Estructura VGGNet para detectar señales de tráfico 63

Figura 4.18 Matriz de confusión 66

Figura 4.19 Imágenes nuevas usadas para comprobar el rendimiento de la CNN 67

Figura 4.20 Resultados del reconocimiento de nuevas imágenes por la CNN 68

Figura 4.21 Resultados del reconocimiento de nuevas imágenes por la CNN 69

Figura 5.1 Intel RealSense D435 72

Figura 5.2 Imagen RGB y de profundidad obtenida por la cámara RealSense 72

Figura 5.3 Nube de puntos obtenido mediante la cámara RealSense 73

Figura 5.4 Ejemplo de una imagen piramidal 74

Figura 5.5 Flujograma del sistema de detección de señales 76

Figura 5.6 Pruebas del sistema de detección de señales en laboratorio 77

Figura 5.7 Reconocimiento señal de stop 78

Figura 5.8 Reconocimiento señal de velocidad 78

Figura 5.9 Reconocimiento señal de prioridad 78

Figura 5.10 Reconocimiento señal de ceda el paso 78

Figura 5.11 Reconocimiento señal de niños cruzando 78

Figura 5.12 Reconocimiento señal prohibido el paso 78

CAPÍTULO 1

INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción

Este capítulo tiene como finalidad el planteamiento del presente Trabajo Fin de Master, los objetivos que con él se persiguen y una breve descripción de los diferentes capítulos que conforman esta memoria.

Dicen que el sentido de la vista fue una de las mayores dudas de Darwin al apostar por su teoría de la evolución. No fue hasta que encontró especies marinas con ojos menos evolucionados cuando realmente se convenció de su propia hipótesis. Para el homosapiens, la vista es el principal mecanismo de obtención de información del entorno. Como humanos percibimos la estructura tridimensional del mundo que nos rodea con facilidad. Prueba de ello lo constatamos al mirar a nuestro alrededor. Somos capaces de identificar los objetos que nos rodean y distinguir los puntos que pertenecen a un objeto de los que corresponden al entorno. Psicólogos han dedicado décadas a intentar entender cómo funciona nuestro sistema visual para poder aplicarlo a las maquinas. La solución al puzle sigue incompleta.

¿Por qué la visión es algo tan complicado? Debemos distinguir entre visión y percepción. Cualquier cámara del mundo es capaz de ver el mundo exactamente como nosotros, pero no es capaz de interpretarlo. En parte, es debido a que la visión es un problema inverso. Intentamos encontrar las incógnitas que nos den la información para solucionar el problema por completo y por ello recurrimos a modelos físicos y de probabilidad para diferenciar entre las posibles soluciones.

La motivación detrás de este trabajo es llegar a entender parte de la tecnología que consigue que un coche sea autónomo, esta tecnología me resulta fascinante, concretamente la parte de visión artificial. En los últimos años se ha producido un avance espectacular en el campo de conducción autónoma teniendo ya a la venta vehículos que son capaces de mantenerse en el carril, tomar el control en un atasco, adaptar la velocidad a la señalada en la vía o al vehículo que le precede. No cabe ninguna duda que de aquí a unos pocos años tendremos circulando en nuestras calles vehículos totalmente autónomos que no necesiten la intervención humana.

Uno de los procesos esenciales para conseguirlo es poder describir el entorno. Llegar al nivel de visión de los humanos es muy complicado, pero gracias al avance de la tecnología con ordenadores cada vez más potentes, se están consiguiendo sistemas muy robustos, que incluso en un futuro podrán ser capaces de superar a la visión humana.

1.2 Objetivos del proyecto

Para el humano el sentido de la vista es el principal mecanismo de obtención de información del entorno, pero cuando se habla de máquinas una de las principales formas de obtener información es mediante cámaras. Una vez obtenida la información de entorno con fotos o videos, tras un procesado de los mismos se puede llegar a identificar los objetos.

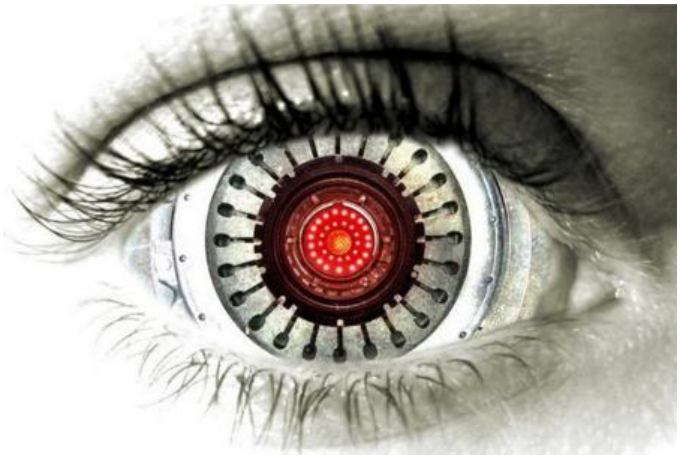


Figura 1.1 El ojo de una maquina

En el presente trabajo se estudia cómo hacer que una maquina identifique objetos, es decir, mediante algoritmos usando los datos obtenidos por una cámara y tras un procesado llevado a cabo por un ordenador, se puede definir en la imagen tomada el objeto que se encuentra en ella. Este proyecto se centrará en identificar líneas viales y señales de tráfico, el cual pueda ser integrado en un sistema de conducción autónoma.

En el mercado, ya se pueden encontrar coches que equipan funciones de este tipo, algunos de ellos incorporan un nivel 2 de conducción autónoma [1] en la figura 1.2 se describen los diferentes niveles existentes. En el nivel 2 es necesario que una persona siempre este atenta a lo que está sucediendo en su entorno, no se espera que hasta dentro de 5 o 10 años se consiga llegar a un nivel 5 en el que la autonomía del vehículo es total y no es necesaria la supervisión de un humano.

Figura 1.2 Niveles de conducción autónoma

La motivación detrás del proyecto es llegar a entender cómo funcionan este tipo de sistemas, ver su complejidad e intentar replicarlo para un sistema de conducción autónoma, con el presente trabajo se podrá hacer una idea de la complejidad que tienen los coches autónomos. Este tipo de vehículos llegará en algún momento al mercado, pero hasta entonces será necesario un largo proceso de investigación.

1.3 Objetivos

El objetivo del trabajo es diseñar e implementar algoritmos capaces de identificar líneas viales y señales de tráfico para aplicación en conducción autónoma. Los objetivos que se han seguido para lograr este fin son:

1) Documentación y búsqueda de información acerca de métodos de detección de objetos mediante visión artificial y métodos de clasificación.

2) Selección de los objetos a analizar.

Sera necesario un conjunto de imágenes para los procesos de detección y clasificación que serán obtenidos a través de cámaras o la web.

3) Programación de software. Se seleccionará una herramienta adecuada (MatLab, LAbVIEW, C#, OpenCV, etc.) para el desarrollo de los algoritmos de visión por computador y aprendizaje de máquina.

Se estudiarán los diferentes algoritmos capaces de detectar objetos con la cámara viendo cual resulta más adecuado para el objetivo del proyecto.

4) Implementación del sistema de detección de señales de tráfico.

5) Realizar pruebas del software y comprobar que se han cumplido los objetivos.

Conjunto de resultados obtenidos y conclusiones llevadas a cabo en el proyecto.

1.4 Descripción de los capítulos

El Trabajo Fin de Master se estructura en los siguientes capítulos.

1er Capítulo: Introducción y objetivos.

En este capítulo se introducirá el trabajo, se hablará de la motivación que ha llevado a su realización, de los objetivos que se persiguen y su estructuración.

2^{o} Capítulo: Estado del arte.

El objetivo de este capítulo es la presentación de la técnica que conforma el trabajo. Se mostrarán los diferentes algoritmos que pueden ser usados, así como las diferentes técnicas de detección de objetos y se hará una introducción a los sistemas de aprendizaje profundo.

3er Capítulo: Sistema de detección de líneas viales.

En este capítulo se verán diferentes formas con las que es posible detectar de líneas viales, como se verá dependiendo de las circunstancias pueden ser usados diferentes métodos, desde simple algoritmos de visión hasta complejos sistemas de aprendizaje profundo.

4º Capítulo: Sistemas de detección se señales de trafico

El problema de detección de señales es complejo debido a las diferentes clases que se pueden encontrar en las carreteras por ello una de las soluciones que se han adoptado es el uso de redes de aprendizaje convolucional. En este capítulo se introducirán este tipo de redes, se explicará el proceso necesario para su entrenamiento, así como los resultados obtenidos.

5º Capítulo: Implementación del sistema de detección de señales

Se diseñará un algoritmo que sea capaz de obtener información del entorno mediante una cámara y analizar la imagen obtenida para comprobar si se encuentra en ella una señal de tráfico y de qué tipo de señal de trata.

6º Capítulo: Conclusiones y trabajos futuros.

Se presentarán las conclusiones que se han obtenido en el proyecto, se propondrán mejoras para el proceso y por último se hablara de posibles trabajos que puedan ser realizados para continuarlo en el futuro.

Bibliografía.

Conjunto de manuales, libros, artículos científicos y sitios web consultados para la elaboración tanto de la parte práctica como de la escritura de la memoria.

CAPÍTULO 2

ESTADO DEL ARTE

2.1 Introducción

En este capítulo se explica en profundidad los fundamentos teóricos necesarios para la elaboración del proyecto. Así, en los siguientes apartados se va a detallar diferentes conceptos de visión artificial que pueden ser usados para la detección de objetos, los cuales se verán en mayor profundidad los usados en el trabajo. Además, se introducirán los sistemas de clasificación como el aprendizaje profundo.

2.2 Fundamentos

La visión artificial también conocida como visión por computador es un subcampo de la de la inteligencia artificial que tiene como finalidad la extracción de información del mundo físico a partir de imágenes.

Uno de los primeros que empezó su estudio fue Larry Roberts[2], que en 1961 creo un programa que podía "ver" una estructura de bloques, analizar su contenido y reproducirlo desde otro perspectiva.

La visión, tanto para un hombre como para un ordenador, consta principalmente de dos fases: captar una imagen e interpretarla. A pesar de la complejidad que presenta el ojo humano, la fase de captación de imágenes hace mucho tiempo que está resuelta. El ojo del ordenador es la cámara de vídeo, y su retina un sensor que es sensible a la intensidad luminosa. Así que en la visión artificial lo complicado es interpretar las imágenes, distinguir los objetos de la escena, extraer información de ellos y resolver aspectos más particulares según las necesidades que se deseen satisfacer.

Actualmente la visión artificial está en fase de crecimiento. Dada la enorme complejidad de ésta, se van solucionando, por etapas, problemas cada vez más complicados. Actualmente, las líneas de estudio e investigación se dividen en múltiples campos, desde la visión tridimensional, a la segmentación y agrupamiento de múltiples objetos diferentes en entornos no controlados, etc.

2.2.1 Formación y representación de la imagen

Una imagen es una representación visual de un objeto iluminado por una fuente radiante. Las que se perciben en las actividades visuales cotidianas provienen normalmente de la luz reflejada por los objetos. La naturaleza básica de una imagen, representada por $f(x,y)$, está caracterizada por dos componentes: la cantidad de luz incidente que procede de la fuente de la escena contemplada; y la cantidad de luz reflejada por los objetos de la escena. Dichas componentes reciben el nombre de iluminación y reflectancia, notándose $i(x,y)$ y $r(x,y)$ respectivamente. Ambas funciones se combinan como producto para dar $f(x,y)$.

En el proceso de formación de la imagen intervienen los siguientes elementos: el objeto, la fuente radiante y el sistema de formación de la imagen que consiste, básicamente, en un sistema óptico, un sensor y un digitalizador. La imagen digital [3] puede ser representada por una matriz f de dimensiones NxM de la forma:

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,M) \\ f(2,1) & f(2,2) & \cdots & f(2,M) \\ \vdots & \vdots & \cdots & \vdots \\ f(N,1) & f(N,2) & \cdots & f(N,M) \end{bmatrix} \quad (1)$$

donde cada elemento, que es el pixel, da la intensidad de la imagen en ese punto.

2.3 Etapas en un proceso de visión artificial

El primer paso en el proceso es adquirir la imagen digital [4], para ello se necesitan sensores y la capacidad para digitalizar la señal producida por el sensor.

Una vez que la imagen digitalizada ha sido obtenida se procesa, esto se realiza para tener mayores posibilidades de éxito a la hora de interpretarla, para ello se realizan una serie de procedimientos.

Un procedimiento que se suele usar es la segmentación, cuyo objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman. En general la segmentación autónoma es uno de los problemas más difíciles en el procesamiento de la imagen. Por una parte, una buena segmentación facilitará mucho la solución del problema; por otra, la segmentación errónea conducirá al fallo.

La salida del proceso de segmentación es una imagen de datos que, o bien contienen la frontera de la región o los puntos de ella misma. Es necesario convertir estos datos a una forma que sea apropiada para el ordenador. La primera decisión es saber si se va a usar la representación por frontera o región completa. La representación por la frontera es apropiada cuando el objetivo se centra en las características de la forma externa como esquinas o concavidades y convexidades. La representación por regiones es apropiada cuando la atención se centra en propiedades internas como la textura o el esqueleto. Sin embargo, en muchas aplicaciones ambas representaciones coexisten.

La elección de una representación es sólo una parte de la transformación de los datos de entrada. Es necesario especificar un método que extraiga los datos de interés. La parametrización, que recibe también el nombre de selección de características que se dedica a extraer rasgos que producen alguna información cuantitativa de interés o características que son básicos para diferenciar una clase de objetos de otra.

En último lugar se encuentran el reconocimiento y la interpretación. El reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los descriptores (clasificación). La interpretación lleva a asignar significado al conjunto de objetos reconocidos.

El proceso descrito puede verse en forma de flujograma en la figura 2.1, esta sería una de las muchas formas que existen para detectar objetos mediante el uso de algoritmos de visión artificial.

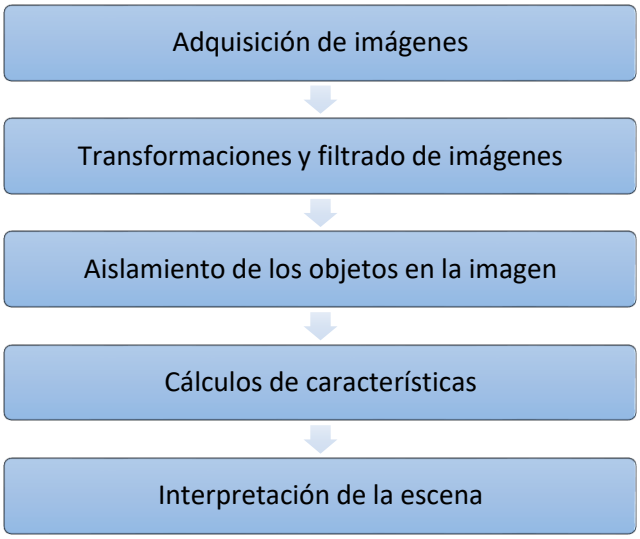


Figura 2.1 Diagrama de bloques de las etapas de un sistema de visión artificial

En los próximos apartados, se van a exponer algunos de los procesos que se usan en las diferentes etapas del flujograma.

2.4 Espacio de color

Cuando vemos un color en la pantalla, lo que en realidad estamos viendo son miles de píxeles que brillan con una cierta intensidad. Cada píxel está formado por tres luces: una roja, una verde y una azul. La sensación de color se produce variando la intensidad de estas tres luces y alterando así la cantidad de luz roja, azul y verde que reciben los ojos del usuario.

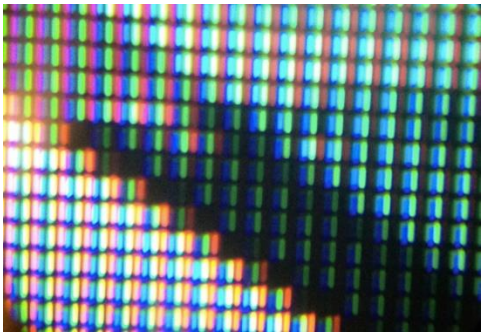


Figura 2.2 Pixeles de una pantalla

En la computadora, los colores vienen codificados por números. Hay varias formas de codificar los colores. La más popular es el sistema de color RGB, que sin duda todos conocéis. Este sistema asigna a cada color una cantidad de Rojo, Verde y Azul entre 0 y 255. Por ejemplo, el rojo puro es [255,0,0]. Hasta aquí todo es lo que se conoce popularmente.

Pero hay otras formas de codificar los colores. El **HSV** [5] (del inglés Hue, Saturation, Value o en español Tono, Saturación y Valor) es ideal para reconocimiento de colores. La definición de cada uno de estos parámetros es:

- **Hue (H)** es el tono de color (por ejemplo, verde, rojo, morado…)

- **Saturation (S)**es la intensidad de esta tonalidad. Cuanta menos saturación, más gris es el color.
- **Value** (**V**) es la luminosidad del color**.**

Es común que se desee elegir un color adecuado para el uso en algún proyecto, cuando es así resulta muy útil usar el espacio de color HSV. En ella el matiz se representa por una región circular; una región triangular separada, puede ser usada para representar la saturación y el valor del color. Normalmente, el eje horizontal del triángulo denota la saturación, mientras que el eje vertical corresponde al valor del color. De este modo, un color puede ser elegido al tomar primero el matiz de una región circular, y después seleccionar la saturación y el valor del color deseados de la región triangular.

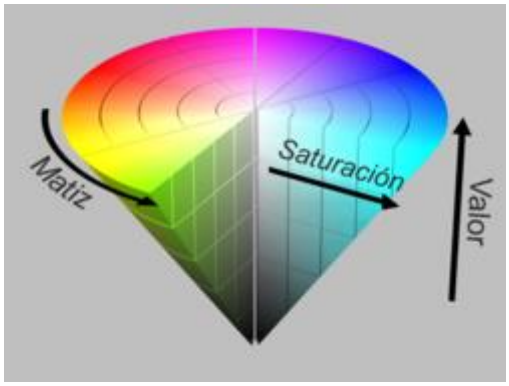


Figura 2.3 Cono del modelo HSV

Matemáticamente se puede realizar la transformación de RGB a HSV como se muestra en las ecuaciones (2), (3) y (4).

$$H = \begin{cases} 0, & if\ MAX = MIN \\ 60 \times \dfrac{G-B}{MAX-MIN} + 360 & if\ MAX = R \\ 60 \times \dfrac{G-B}{MAX-MIN} + 120 & if\ MAX = G \\ 60 \times \dfrac{G-B}{MAX-MIN} + 240 & if\ MAX = B \end{cases} \tag{2}$$

$$V = \frac{1}{2}(MAX + MIN) \tag{3}$$

$$S = \begin{cases} 0, & if\ MAX = MIN \\ \dfrac{MAX-MIN}{MAX-MIN} = \dfrac{MAX-MIN}{2L} & if \leq \dfrac{1}{2} \\ \dfrac{MAX-MIN}{2-(MAX-MIN)} = \dfrac{MAX-MIN}{2-2L} & if\ \leq \dfrac{1}{2} \end{cases} \tag{4}$$

2.5 Detección de bordes

En visión artificial, la detección de los bordes en una imagen es una parte importante y de gran utilidad, pues facilita muchas tareas, entre ellas, el reconocimiento de objetos y la segmentación de regiones, entre otras.

2.5.1 Algoritmo de Canny

El algoritmo de Canny [6], basado en la primera derivada, está considerado como uno de los mejores métodos de detección de contornos mediante el empleo de máscaras de convolución. Los puntos de contorno son como zonas de píxeles en las que existe un cambio brusco de nivel de gris. En el tratamiento de imágenes, se trabaja con píxeles, y en un ambiente discreto.

El algoritmo de Canny [7] consiste en tres grandes pasos.

1. Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.
2. Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.
3. Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

2.5.1.1 Obtención del gradiente

Para la obtención del gradiente, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el objetivo de suavizar la imagen y tratar de eliminar el posible ruido existente. Sin embargo, se debe de tener cuidado de no realizar un suavizado excesivo, pues se podrían perder detalles de la imagen y provocar un pésimo resultado final. Una vez que se suaviza la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente, obteniendo así dos imágenes. El algoritmo para este primer paso se describe a continuación.

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\frac{1}{115}$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figura 2.4 Máscaras de convolución para el obtener el filtro gaussiano

2.5.1.2 Supresión no máxima al resultado del gradiente

Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes adelgazados. El procedimiento es el siguiente: se consideran cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje

horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente.

1. Suavizar la imagen I mediante un filtro gaussiano y obtener J como imagen de salida.
2. Para cada píxel (i, j) en J, obtener la magnitud y orientación del gradiente basándose en las siguientes expresiones:
   El gradiente de una imagen f(x,y) en un punto (x,y) se define como un vector bidimensional dado por las ecuaciones (5) y (6).

$$G_x(x,y) = I(x+1,y) - I(x-1,y) \tag{5}$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1) \tag{6}$$

   siendo un vector perpendicular al borde, donde el vector G apunta en la dirección de variación máxima de f en el punto (x,y) por unidad de distancia, con la magnitud y dirección dadas por las ecuaciones (7) y (8).

$$M_G(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \tag{7}$$

$$\theta_G(x,y) = \tan^{-1}\left(\frac{G_x(x,y)}{G_y(x,y)}\right) \tag{8}$$

3. En el último paso se normaliza el vector de características para hacerlos más robustos antes cambios de iluminación, para ello se usa la ecuación (9).

$$v = \frac{v}{\sqrt{\epsilon + \|v\|_2^2}} \tag{9}$$

   Donde ϵ=0,001 sirve para evitar divisiones por cero.

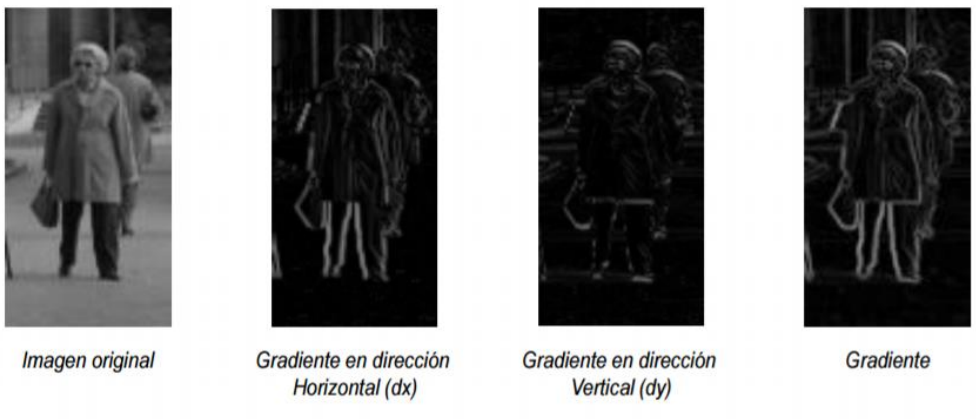


Figura 2.5 Gradiente en dirección horizontal y vertical de una imagen

Posteriormente se observa si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. De ser así se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

2.5.1.3 Histéresis de umbral a la supresión no máxima

La imagen obtenida en el paso anterior suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral. El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo. Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos local conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayo res al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. Es así como en este paso se logra eliminar las uniones en forma de Y de los segmentos que confluyan en un punto.

Finalmente se puede ver el resultado en la figura 2.6.

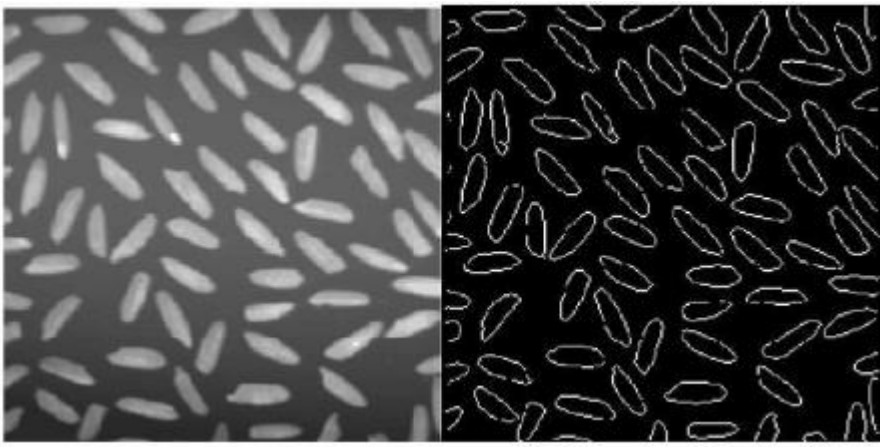


Figura 2.6 Resultado de la detección de bordes mediante el algoritmo de Canny

2.5.2 Operador Sobel

Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. Para cada punto de la imagen a procesar, el resultado del operador Sobel es tanto el vector gradiente correspondiente como la norma de este vector.

El operador Sobel[8] calcula el gradiente de la intensidad de una imagen en cada punto (píxel). Así, para cada punto, este operador da la magnitud del mayor cambio posible, la dirección de éste y el sentido desde oscuro a claro. El resultado muestra cómo de abruptamente o suavemente cambia una imagen en cada punto analizado y, en consecuencia, cuán probable es que éste represente un borde en la imagen y, también, la orientación a la que tiende ese borde. En la práctica, el cálculo de la magnitud -probabilidad de un borde- es más fiable y sencillo de interpretar que el cálculo de la dirección y sentido.

Matemáticamente, el gradiente de una función de dos variables (en este caso, la función de intensidad de la imagen) para cada punto es un vector bidimensional cuyos componentes están dados por las primeras derivadas de las direcciones verticales y horizontales. Para cada punto de la imagen, el vector gradiente apunta en dirección del incremento máximo posible de

la intensidad, y la magnitud del vector gradiente corresponde a la cantidad de cambio de la intensidad en esa dirección.

Lo dicho en los párrafos anteriores implica que el resultado de aplicar el operador Sobel sobre una región con intensidad de imagen constante es un vector cero, y el resultado de aplicarlo en un punto sobre un borde es un vector que cruza el borde (perpendicular) cuyo sentido es de los puntos más oscuros a los más claros.

Matemáticamente, a diferencia del algoritmo de Canny el operador Sobel utiliza dos kernels de 3×3 elementos para aplicar convolución a la imagen original para calcular aproximaciones a las derivadas, un kernel para los cambios horizontales y otro para las verticales. Si definimos A como la imagen original, el resultado, que son las dos imágenes G_x y G_y que representan para cada punto las aproximaciones horizontal y vertical de las derivadas de intensidades, es calculado por las ecuaciones (10) y (11).

$$G_x(x,y) = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \tag{10}$$

$$G_y(x,y) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \tag{11}$$

En cada punto de la imagen, los resultados de las aproximaciones de los gradientes horizontal y vertical pueden ser combinados para obtener la magnitud del gradiente y la la dirección del gradiente, mediante las ecuaciones (12) y (13).

$$M_G(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \tag{12}$$

$$\theta_G(x,y) = \tan^{-1}\left(\frac{G_x(x,y)}{G_y(x,y)}\right) \tag{13}$$

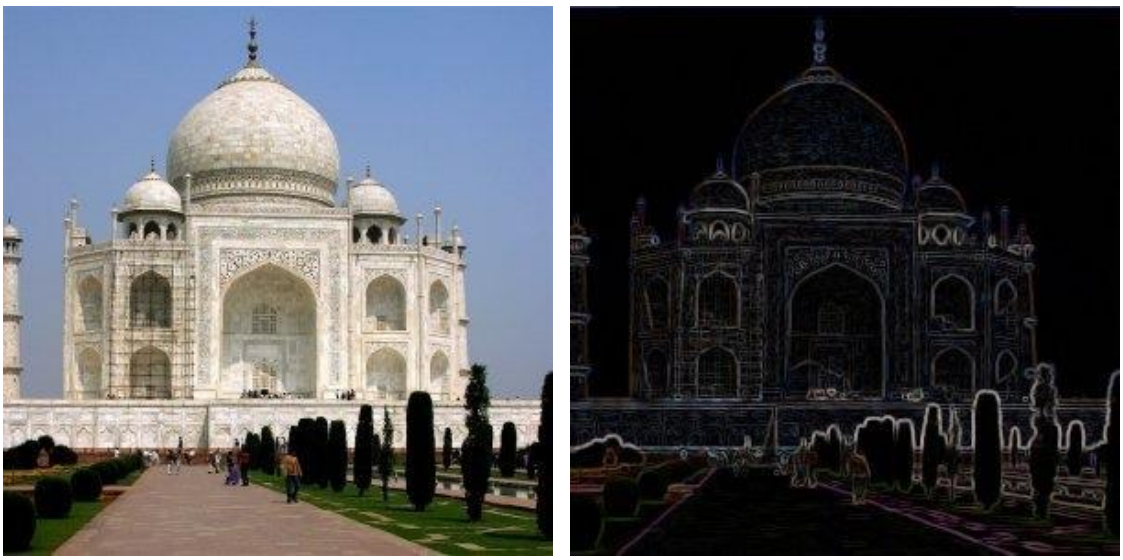


Figura 2.7 Uso operador Sobel

2.6 Transformada de Hough

La transformada de Hough [9] es utilizada para detectar líneas en aplicaciones de procesamiento de imágenes, para realizar lo anterior, transforma los puntos del plano cartesiano a un espacio de parámetros. Se pueden clasificar la transformada en dos grandes grupos dependiendo de la parametrización usada para representar líneas.

El primero utiliza los parámetros de la pendiente a y del parámetro de intersección con la ordenada b, para lo cual utiliza la representación de la recta dada por la ecuación (14).

$$y = a \cdot x + y \tag{14}$$

Por cada punto del plano pasan infinitas rectas, que satisfacen la ecuación anterior para diferentes valores de a y b. Considerando el plano ab denominado espacio de parámetros, da una única recta para cada punto, por ejemplo, si consideramos dos puntos (xi,yi) y (xj,yj), cada punto tendrá asociada una única recta las cuales al interceptarse darán los parámetros de a y b de la recta que contiene los puntos colineales de (xi,yi) y (xj,yj), tal como se muestra en la siguiente figura 2.8.

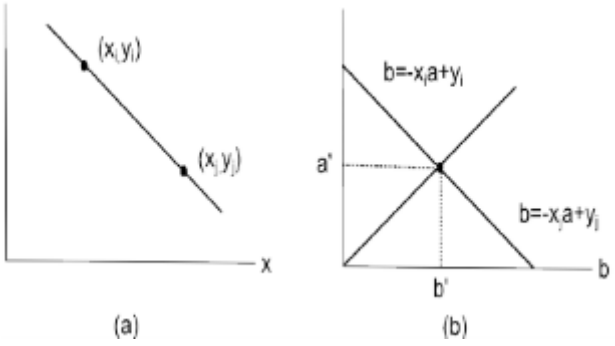


Figura 2.8 (a) Recta en el plano xy; (b) Espacio de parámetros.

En la implementación de la transformada se utilizan unas celdas acumuladoras, las cuales dividen el espacio de parámetros a fin de evaluar todos los posibles valores por cada punto e ir acumulando los frecuentes (fase de votación), dando como resultado la creación de una malla en los cuales mientras mayor sea el valor en cada casilla significa que se ha encontrado una línea recta (fase de localización de picos en la malla) [8]. El inconveniente de utilizar el espacio de parámetros es la dificultad de detectar líneas verticales debido a que los parámetros pueden llegar a ser infinitos mientras la línea se hace vertical.

El segundo grupo utiliza coordenadas polares y está compuesto por los parámetros ρ y θ, como se expresa en la ecuación (15).

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta \tag{15}$$

Donde ρ es la distancia de la línea con el origen y θ es el ángulo entre el vector con respecto al eje de las abscisas, el comportamiento es similar a lo realizado en el primer grupo, la diferencia radica que, en vez de rectas, por cada punto estará asociado una sinusoide en el plano ρ θ, como se muestra en la figura. Utilizando las coordenadas polares se soluciona el inconveniente del primer grupo.

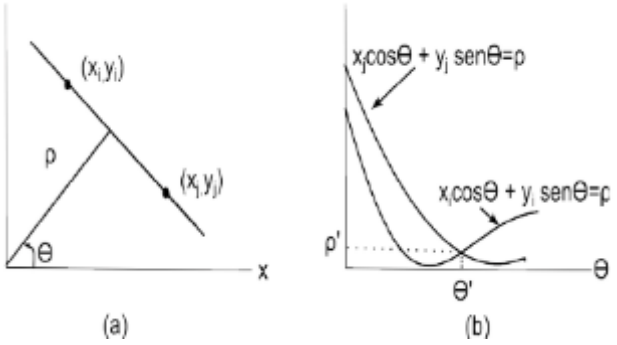


Figura 2.9 (a) Parametrización de las líneas en el plano xy; (b) Curvas sinusoidales en el plano ρ ϑ

2.7 Conceptos de inteligencia artificial

Hoy en día todo el mundo habla de Machine Learning, Deep Learning o inteligencia artificial (aunque se suelen utilizar los términos en inglés, en español seria aprendizaje automático y aprendizaje profundo), miles de empresas alrededor del mundo dicen usarla.

Pero en que se diferencias cada una de ellas, estos términos se van a estudiar en profundidad.

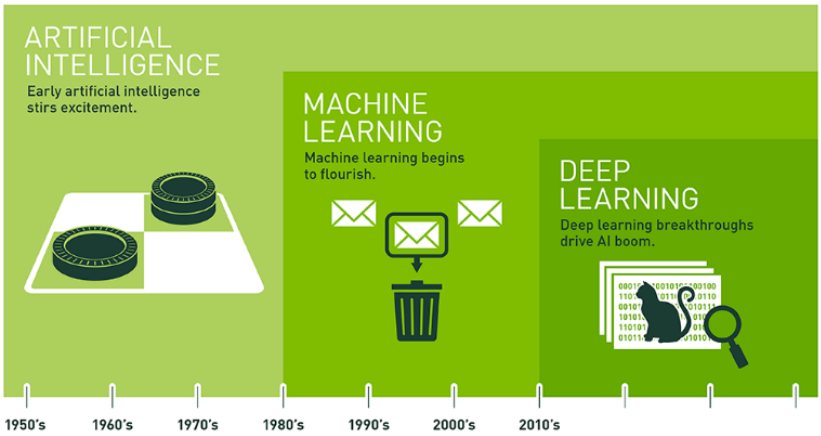


Figura 2.10 Etapas de la inteligencia Artificial

La Inteligencia Artificial (IA) es la combinación de algoritmos con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano. Una tecnología que todavía nos resulta lejana y misteriosa, pero que desde hace unos años está presente en nuestro día a día a todas horas. Con la IA se resuelven numerosos problemas de manera "inteligente", pero estos se deben encontrar en un campo muy concreto y preciso, por ejemplo, un algoritmo que sea capaz de detectar si el iris del ojo humano pertenece a una mujer o un hombre, la maquina es capaz de actuar, percibir y razonar como lo haría un ser humano, pero única y exclusivamente para ese problema en concreto, que es para el que ha sido entrenada.

2.7.1 Aprendizaje automático

Arthur Samuel [10] definió el aprendizaje automático como el campo de estudio que le da a las maquinas la habilidad de aprender sin ser específicamente programas.

En su uso más básico es la práctica de usar algoritmos para usar datos, aprender de ellos y luego ser capaces de hacer una predicción o sugerencia sobre algo. Los programadores con el aprendizaje automático buscan algoritmos para convertir muestras de datos en programas de computadora, sin tener que escribir los últimos explícitamente. Los modelos o programas resultantes deben ser capaces de generalizar comportamientos para un conjunto más amplio (potencialmente infinito) de datos que no haya sido usado para su entrenamiento.

Desde los primeros albores de la temprana inteligencia artificial, los algoritmos han evolucionado con el objetivo de analizar y obtener mejores resultados: árboles de decisión, programación lógica inductiva (ILP), clustering para almacenar y leer grandes volúmenes de datos, redes Bayesianas y un numeroso abanico de técnicas que los programadores de data science pueden aprovechar.

Los diferentes algoritmos de aprendizaje automático se agrupan en función de la salida de los mismos, estos se dividen en tres grandes grupos.

- Aprendizaje supervisado: Es el aprendizaje “con maestro”, es decir, para cada ejemplo se indica cual es la solución.
- Aprendizaje por refuerzo: En este caso no se indica la solución en cada ejemplo, pero si se indica si está bien o mal.
- Aprendizaje no supervisado: Todo el proceso se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas.

En la última década los sistemas de aprendizaje automático, han sufrido una gran evolución debido a los avances de hardware, ya que con los equipos actuales se pueden hacer cosas que antiguamente eran inimaginables. Con las capacidades de cálculo de los actuales computadores ha generalizado el uso de técnica conocida como aprendizaje profundo.

2.7.2 Aprendizaje profundo

El aprendizaje profundo [11] lleva a cabo el proceso de aprendizaje automático usando una red neuronal artificial que se compone de un número de niveles jerárquicos. En el nivel inicial de la jerarquía la red aprende algo simple y luego envía esta información al siguiente nivel. El siguiente nivel toma esta información sencilla, la combina, compone una información algo un poco más compleja, y se lo pasa al tercer nivel, y así sucesivamente.

Así es el aprendizaje profundo, redes neuronales gigantescas que tiene gran poder de aprendizaje, hace años esto sería impasable pues para llegar a algo así se necesitarían décadas para alcanzarlo. Hoy en día gracias a algoritmos diseñados con grandes cantidades de datos se han obtenido resultados increíbles como las redes GAN [12] (Generative Adversarial Networks) o el algoritmo YOLO [13] (You Only Look Once).

Se ha abierto un amplio abanico de posibilidades y día tras día estamos viendo esas mejoras en las conferencias por todo el mundo como aplicaciones que usamos en nuestra vida cotidiana. A continuación, se va a mostrar cómo funciona una red neuronal.

2.8 Redes neuronales

Las primeras investigaciones sobre el tema, datan aproximadamente de principios del siglo XIX, pero no fue hasta la década de los años 40 y 50, en el siglo XX, cuando cobró mayor fuerza, gracias al movimiento Conexionista. Este movimiento sostenía la premisa de que el secreto del proceso del aprendizaje y el conocimiento se halla fundamentalmente en axiomas o verdades incuestionables y que el conocimiento es independiente de la estructura que manejen los símbolos, y la representación del conocimiento se hace desde el estrato más básico de la inteligencia: el cerebro, especialmente en las neuronas y las múltiples conexiones entre ellas.

Este interés notable por las redes neuronales se vio disminuido en los años 70 debido al surgimiento de autores como Minsky y Papert (1969) los cuales manifestaron las limitaciones del proceso de aprendizaje de las arquitecturas de las redes neuronales.

El principio central del conexionismo es que los fenómenos mentales pueden ser descritos por redes de unidades sencillas y frecuentemente iguales que se interconectan.

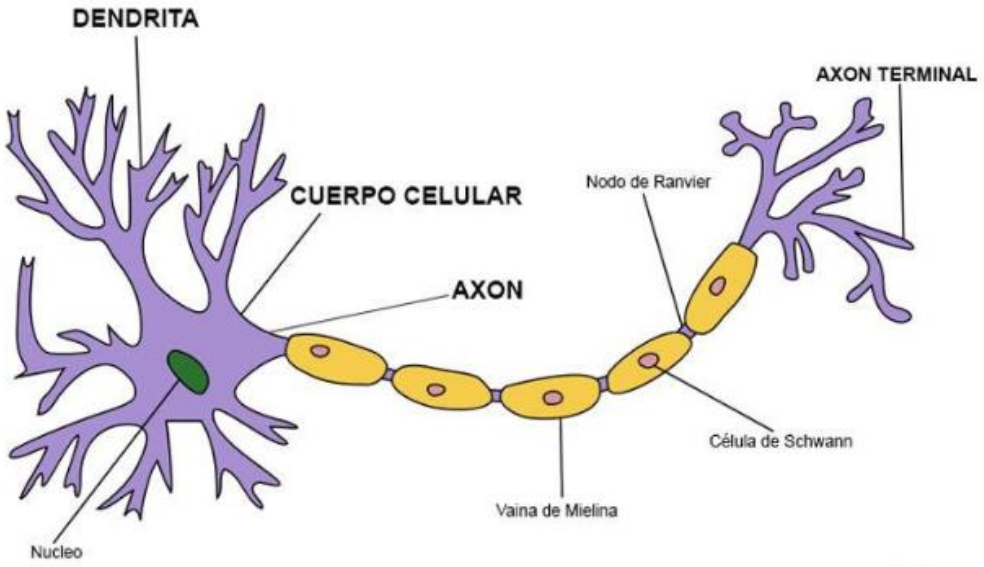


Figura 2.11 Estructura de una neurona

La neurona [14], como la que se puede ver en la figura 2.11, es la unidad fundamental el sistema nervioso y en particular del cerebro. Cada neurona es una simple unidad procesadora que recibe y combina señales desde y hacia otras neuronas. Si la combinación de entradas es suficientemente fuerte la salida de la neurona se activa.

El cerebro consiste en uno o varios billones de neuronas densamente interconectadas. El axón (salida) de la neurona se ramifica y está conectada a las dendritas (entradas) de otras neuronas a través de uniones llamas sinapsis. La eficacia de la sinapsis es modificable durante el proceso de aprendizaje de la red.

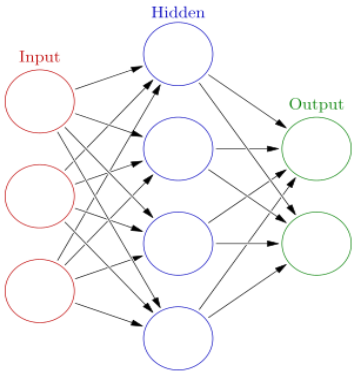


Figura 2.12 Esquema de una red neuronal

En la figura 2.12, cada nodo circular representa una neurona artificial y cada flecha representa una conexión desde la salida de una neurona a la entrada de otra. El principal objetivo de una neurona artificial es resolver problemas del mismo modo que lo haría un cerebro humano, aunque las redes neuronales son bastante más abstractas.

Un aspecto interesante de estos sistemas es que son impredecibles en su éxito con el auto-aprendizaje. Después del entrenamiento algunos se vuelven expertos en la resolución de un problema y otros en cambio no funcionan tan bien.

Las RNA (Redes Neuronales Artificiales) están inspiradas en las redes neuronales biológicas del cerebro humano. Están constituidas por elementos que se comportan de forma similar a una neurona biológica en sus funciones más comunes. Dichos elementos están organizados de una forma similar a como se presentan en el cerebro.

2.8.1 Redes Neuronales Artificiales

Las RNA al margen de "parecerse" al cerebro presentan una serie de características propias de él, como es la capacidad de aprender de la experiencia a través de ejemplos, para ello abstraen las características principales. Están inspiradas en la biología, lo cual significa que están formadas por elementos que se comportan de manera análoga a las neuronas (en las funciones más elementales) y están organizadas de una forma similar a la del cerebro humano, pero las analogías no son muchas más.

Las características fundamentales de las RNA son:

- Aprenden de la experiencia: Las RNA pueden modificar su comportamiento como respuesta a su entorno. Dado un conjunto de entradas (quizá con las salidas deseadas), las RNA se ajustan para producir respuestas consistentes. Una amplia variedad de algoritmos de entrenamiento se ha desarrollado, cada uno con sus propias ventajas e inconvenientes.
- Generalizan de ejemplos anteriores a los ejemplos nuevos: Una vez que la RNA esté entrenada, la respuesta de la red puede ser, hasta un cierto punto, insensible a pequeñas variaciones en las entradas, lo que las hace idóneas para el reconocimiento de patrones.
- Abstracción de la esencia de las entradas: Algunas RNA son capaces de abstraer información de un conjunto de entradas. Por ejemplo, en el caso de reconocimiento

de patrones, una red puede ser entrenada en una secuencia de patrones distorsionados de una letra. Una vez que la red sea correctamente entrenada será capaz de producir un resultado correcto ante una entrada distorsionada, lo que significa que ha sido capaz de aprender algo que nunca había visto. En las Redes Neuronales Artificiales, ANN, la unidad análoga a la neurona biológica es el elemento procesador, PE (process element). Un elemento procesador tiene varias entradas y las combina, normalmente con una suma básica. La suma de las entradas es modificada por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del elemento procesador.

2.8.1.1 Redes de capa simple

A pesar de que una sola neurona puede realizar modelos simples de funciones, su mayor productividad viene dada cuando se organizan en redes. La red más simple es la formada por un conjunto de perceptrones a los que entra un patrón de entradas y proporcionan la salida correspondiente. Por cada perceptrón que tengamos en la red vamos a tener una salida, que se hallará como se hacía con un perceptrón solo, haciendo el sumatorio de todas las entradas multiplicadas por los pesos. Al representar gráficamente una red, se añade una "capa" inicial que no es contabilizada a efectos de computación, solamente sirve para distribuir las entradas entre los perceptrones. La denominaremos la capa 0.

De esta manera, la representación gráfica de una red de capa simple se puede ver en la figura 2.13.

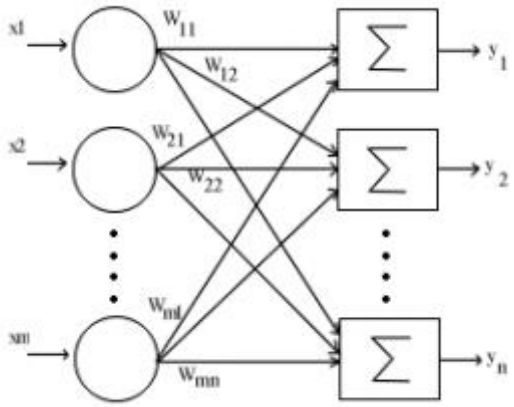


Figura 2.13 Esquema de una Red Neuronal Artificial de capa simple

2.8.1.1 Redes multicapa

Las redes multicapa se forman por un conjunto de redes de capa simple en cascada unidas por pesos, donde la salida de una capa es la entrada de la siguiente capa. Generalmente son capaces de aprender funciones que una red de capa simple no puede aprender, por lo que ofrecen mejores capacidades computacionales. Para que este incremento en poder computacional sea tal, tiene que existir una función de activación no lineal entre las capas, por lo que generalmente se utilizará una función de activación sigmoidea en detrimento de la lineal o umbral.

Para calcular la salida de una red multicapa se debe hacer de la misma manera que en las redes de capa simple, teniendo en cuenta que las salidas de una capa son las entradas de la siguiente capa. La representación gráfica se puede ver en la figura 2.14.

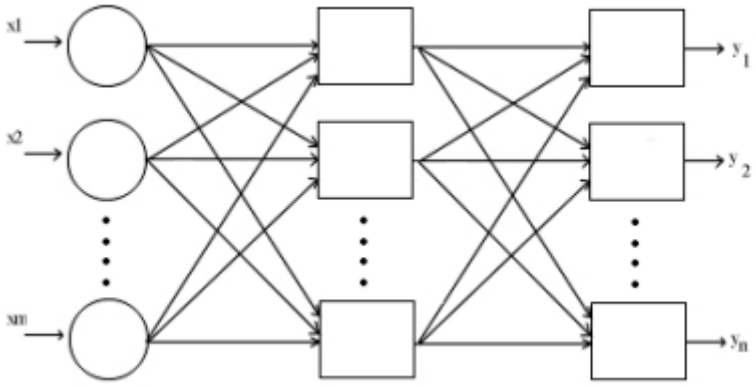


Figura 2.14 Esquema de una Red Neuronal Artificial multicapa

2.8.1.2 Conceptos

Los modelos de redes neuronales artificiales [15], son modelos matemáticos esencialmente simples que definen una función $f:X \rightarrow Y$. Pero a veces los modelos también están íntimamente asociados con un algoritmo de aprendizaje en particular o regla de aprendizaje.

A continuación, se definen algunos conceptos que se usan en los modelos de RNN.

- **Función de red**

La palabra red en el término "red neuronal artificial" se refiere a las interconexiones entre las neuronas en las diferentes capas de cada sistema. Una red neuronal simple tiene tres capas. La primera capa tiene neuronas de entrada que envían datos a través de las sinapsis a la segunda capa de neuronas, y luego a través de más sinapsis a la tercera capa de neuronas de salida. Los sistemas más complejos tendrán más capas, algunos aumentando las de entrada y de salida de neuronas. Las sinapsis almacenan parámetros llamados "pesos" que manipulan los datos en los cálculos.

Un RNA se define típicamente por tres tipos de parámetros:

1. El patrón de interconexión entre las diferentes capas de neuronas.
2. El proceso de aprendizaje para la actualización de los pesos de las interconexiones.
3. La función de activación que convierte las entradas ponderadas de una neurona a su activación a la salida.

Matemáticamente, la función de red de una neurona $f(x)$ se define como una composición de otras funciones $g_i(x)$. Este se representa como una estructura de red, con flechas que representan las dependencias entre variables. Un tipo ampliamente utilizado de la composición es la suma ponderada no lineal, mostrada en la ecuación (16).

$$f(x) = k(\sum w_i g_i(x)) \tag{16}$$

Dónde *k* (denominado comúnmente como la función de activación) es una función predefinida, como la tangente hiperbólica o función sigmoide. Una característica importante de la función de activación es que proporciona una transición suave como valores de entrada de cambio, es decir, un pequeño cambio en la entrada produce un pequeño cambio en la producción.

- **Aprendizaje**

Lo que ha atraído el mayor interés en las redes neuronales es la posibilidad de aprendizaje. Dada una determinada tarea a resolver, y una clase de funciones F, el aprendizaje consiste en utilizar un conjunto de observaciones para encontrar la forma de resolver la tarea de alguna forma óptima.

Esto implica la definición de una función de coste C: F →R, y se puede decir que ninguna solución tiene un costo menor que el costo de la solución óptima.

La función de coste C es un concepto importante en el aprendizaje, ya que representa lo lejos que una solución particular se encuentra de la solución óptima al problema a resolver. Los algoritmos de aprendizaje buscan encontrar una función que tiene el menor costo posible.

- **Elección de función de coste**

Una función de coste J(x) mide que tan insatisfechos estamos con las predicciones del modelo con respecto a una respuesta correcta. Existen varias funciones de coste como el error cuadrático medio o entropía cruzada y la selección de uno de ellos depende de varios factores como el algoritmo seleccionado o el nivel de confianza deseado, pero principalmente depende del objetivo de nuestro modelo.

- **Tipos de entrada**

Finalmente, también se pueden clasificar las RNAs según sean capaces de procesar información de distinto tipo. Las redes analógicas, procesan datos de entrada con valores continuos y, habitualmente acotados. Las redes discretas: procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos booleanos.

2.8.1.3 Aplicaciones

Las redes neurales artificiales son bastante apropiadas para aplicaciones en las que no se dispone a priori de un modelo identificable que pueda ser programado, pero se dispone de un conjunto básico de ejemplos de entrada (previamente clasificados o no). Asimismo, son altamente robustas tanto al ruido como a la disfunción de elementos concretos y son fácilmente paralelizables.

Esto incluye problemas de clasificación y reconocimiento de patrones de voz, imágenes, señales, etc. Asimismo, se han utilizado para encontrar patrones de fraude económico, hacer predicciones en el mercado financiero, hacer predicciones de tiempo atmosférico, etc.

También se pueden utilizar cuando no existen modelos matemáticos precisos o algoritmos con complejidad razonable, por ejemplo, la red de Kohonen ha sido aplicada con un

éxito más que razonable al clásico problema del viajante (un problema para el que no se conoce solución algorítmica de complejidad polinómica).

Otro tipo especial de redes neuronales artificiales se ha aplicado en conjunción con los algoritmos genéticos (AG) para crear controladores para robots. La disciplina que trata la evolución de redes neuronales mediante algoritmos genéticos se denomina Robótica Evolutiva. En este tipo de aplicación el genoma del AG lo constituyen los parámetros de la red (topología, algoritmo de aprendizaje, funciones de activación, etc.) y la adecuación de la red viene dada por la adecuación del comportamiento exhibido por el robot controlado (normalmente una simulación de dicho comportamiento).

Podemos encontrar que este tipo de sistemas son usados para solucionar problemas de la vida real como:

- Aproximación de funciones, o el análisis de regresión, incluyendo la predicción de series temporales, funciones de aptitud y de modelado.
- Clasificación, incluyendo el reconocimiento de patrones y la secuencia de reconocimiento, detección y de la toma de decisiones secuenciales.
- Procesamiento de datos, incluyendo el filtrado, el agrupamiento, la separación ciega de las señales y compresión.
- Robótica, incluyendo la dirección de manipuladores y prótesis.
- Ingeniería de control, incluyendo control numérico por computadora.

Las redes neuronales artificiales se han utilizado también para el diagnóstico de varios tipos de cáncer. Los diagnósticos se pueden utilizar para hacer modelos específicos tomados de un gran grupo de pacientes en comparación con la información de un paciente dado.

CAPÍTULO 3

SISTEMA DE DETECCIÓN DE LÍNEAS VIALES

3.1 Introducción

En este capítulo se describirá el desarrollo de un sistema de detección de líneas viales para aplicación en conducción autónoma, así como se hablará de diferentes técnicas que puede ser usadas para su detección.

Los primeros sistemas de detección de carreteras basados en visión por computador comienzan a aparecer en la década de los 80, y principalmente estaban orientados a la conducción automática. En EEUU destaca el sistema de Carnegie Mellon University (CMU)para el vehículo NavLab [16] y en Europa la Universidad de Bundeswer-Munchen con su sistema VITA para el vehículo VaMoRs [17].

Sin embargo, en la última década están surgiendo cada vez más iniciativas dedicadas al diseño de sistemas de ayuda a la conducción, motivados por las necesitas de obtener sistemas de ayuda a la conducción que permitan reducir la siniestralidad, así como para el uso en vehículos autónomos. Aunque ambos enfoques comparten la misma tecnología, existen ligeras diferencias entre ellos:

- En la conducción automática se extrae únicamente la información necesaria para deducir el giro del volante, la velocidad, etc. El objetivo del sistema de percepción es proporcionar una descripción del entorno que permita la navegación autónoma del vehículo. Puede ser suficiente con estimar la posición y orientación instantánea del vehículo dentro del carril, incluso con un modelo muy sencillo de carretera, pero la fiabilidad de estos datos debe ser muy alta.
- En la asistencia al conductor, los requisitos de percepción son diferentes. Ya no se trata de tomar el control del vehículo, sino de percibir todos los elementos relevantes del entorno para evaluar el potencial riesgo de cada situación. Por lo tanto, es necesario obtener más información del entorno para evaluar la situación y predecir maniobras. Si se está a tiempo de evitar el accidente, el sistema deberá emitir las alertas necesarias para que el conductor ejecute una acción correctora, y si el accidente es inevitable, se deberá preparar el vehículo para minimizar las consecuencias. En general, la investigación en este campo persigue fines comerciales y/o sociales.

3.2 Sistemas de detección de líneas viales

En cualquier escenario de conducción, la detección de las líneas viales es una de las principales maneras de saber dónde dirigir el vehículo. Las carreteras donde estas líneas se pueden identificar con facilidad son más seguras, ya que el conductor sabrá en todo momento donde debe situar al automóvil. A la hora de plantear un vehículo sin conductor es un buen punto de inicio ya que simplemente identificando estas líneas se podría dar órdenes a este para dirigirse a un lugar.

3.2.1 Detección de líneas rectas

En este apartado se desarrolla un sistema de detección de líneas viales que se encuentren en buen estado y no tengan curvas pronunciadas, para ello se usaran diferentes funciones, que se pueden encontrar en la librería de código abierto para visión por computador

OpenCV [18] que fue originalmente desarrollada por Intel . En el desarrollo de este proyecto se ha utilizado la versión de esta librería para el lenguaje de programación Python [19]. Las líneas de carril detectadas serán como las que se muestran en la figura 3.1.

Figura 3.1 Autovía Cartagena-Murcia

El flujograma que se ha seguido para conseguir la detección se puede ver en la figura 3.2.

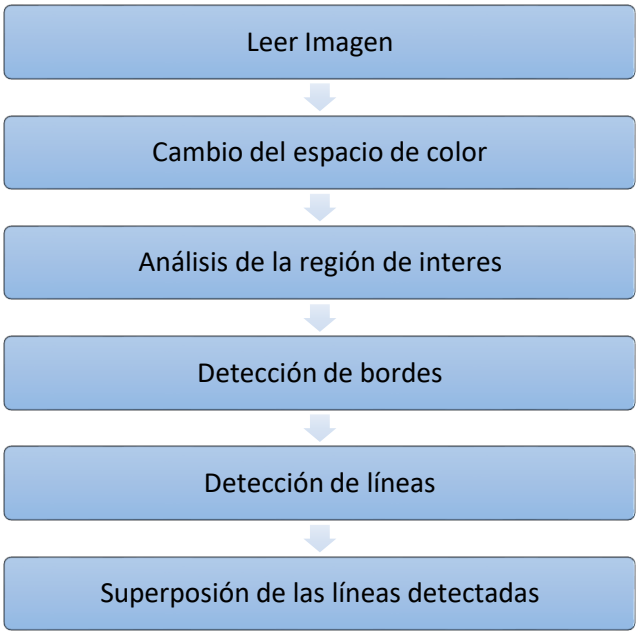


Figura 3.2 Flujograma del sistema de detección de líneas viales

3.2.1.1 Análisis de la imagen

Una vez que han sido obtenidas las imágenes nos podemos encontrar con ciertos problemas que hay que intentar evitar para interpretar la mayor cantidad de información de manera precisa.

- Ruido: Es producido por algún patrón regular, por ejemplo, la presencia de objetos que al momento de procesar dan lugar a formas parecidas a las marcas viales lo que puede confundir al algoritmo como pueden ser marcar como flechas o sombras que se han proyectado en la carretera.

- Oclusión: Obstrucción de las líneas generas por sobras o los vehículos.
- Falla en la pintura asfáltica: muchas veces las carreteras no se encuentran en buen estado y no son visibles las marcas viales.

3.2.1.2 Cambio de espacio de color

Generalmente las líneas viales en España son de color blanco o incluso a veces amarillo, por lo que colores como el verde, azul o marrón pueden ser excluidos, para ello se cambiará el espacio de color de la imagen. Lo que se pretende conseguir es descartar los colores en los que no se puedan encontrar líneas viales, de esta manera toda la información innecesaria será eliminada de la imagen.

Se pretende que los colores blanco y amarillo sean los únicos visibles en la imagen utilizando el espacio de color HSV, para ello se ha limitado el rango de colores visibles. Los pasos que se han seguido son:

1. Para obtener los colores amarillos, se eliminarán los píxeles con un valor de Hue (H) que se encuentre fuera del rango entre 10 y 50 y se usará un valor de saturación (S) alto.
2. Para obtener los colores blancos, se elimina los píxeles que tengan un valor de luminosidad (V) inferior a 190.
3. Una vez realizados estos cambios se obtendrá una imagen donde solo sean visibles las zonas de color blanco o amarrillo.

Se ha realizado el cambio del espacio de color en la figura 3.1, obteniendo el resultado mostrado en la figura 3.3.

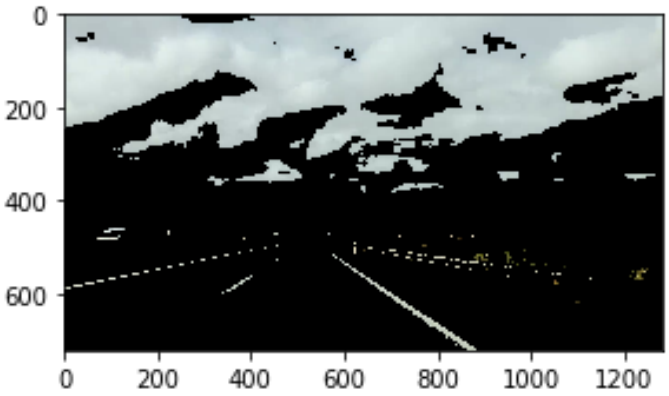


Figura 3.3 Cambio a espacio de color HSV

3.2.1.3 Selección de la región de interés

Tras el cambio del espacio de color, aún se encuentran partes que se han conservado en la imagen, como se observa en la figura 3.3, pero que no son líneas de carril, estas siempre se podrán encontrar en la zona central e inferior de la imagen, por lo que el resto de zonas pueden ser eliminadas al ser información redundante.

La ROI es el conjunto o área de pixeles en la imagen, que se quiere clasificar en clases aplicando diferentes técnicas. Esta zona será de vital importancia a la hora de identificar los objetos y tratar de disminuir los errores.

Si se observa la figura 3.1 se puede ver que la carretera contiene muchos elementos, en primera instancia se puede ver la carretera y en segundo plano vehículos en el carril contrario, el guarda rail, el cielo, arbustos en definitiva objetos que no son necesarios detectar

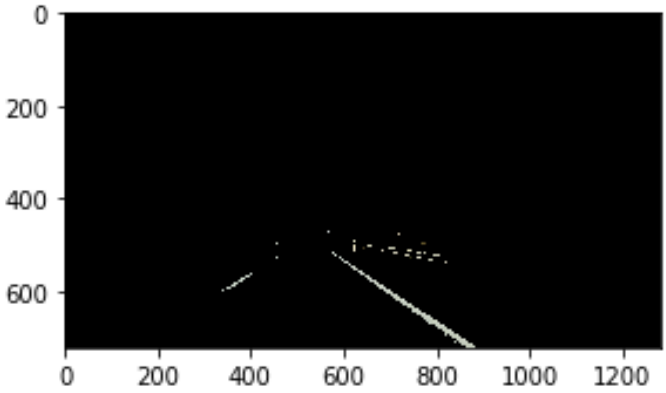


Figura 3.4 Selección de la región de interés

Para realizar la búsqueda se puede observar que se las líneas de carril se encuentran en la zona inferior de la imagen por lo que la zona superior puede ser eliminada. En la figura 3.4 se puede ver con más exactitud, la información que será eliminada se encuentra antes de la columna 200 y la después de la 900 y a partir de la fila 400 de la imagen no se encuentran ninguna información que pueda ser de interés.

3.2.1.4 Algoritmo de Canny

Una vez que se tienen aisladas las imágenes donde se encuentran las líneas de carril, se puede calcular los bordes de manera fácil usando el algoritmo de Canny. Los píxeles cerca de los bordes generalmente tienen un gradiente alto o tasa de cambio de valor así se puede usar esto para detectar líneas de carril.

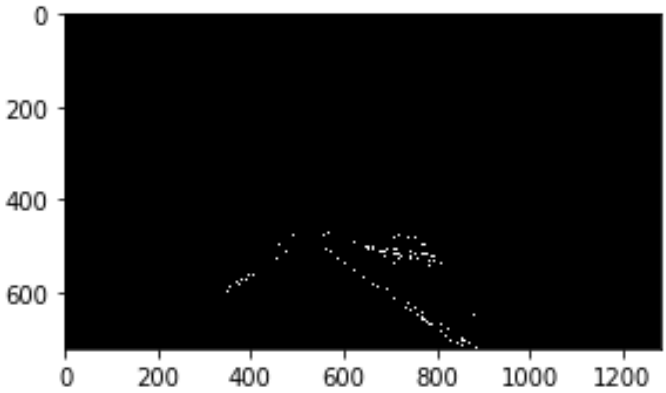


Figura 3.5 Detección de bordes algoritmo de Canny

3.2.1.5 Transformada de Hough

A partir de los bordes obtenidos, se tendrán que calcular las líneas para ello se usara la transformada de Hough, se comenzara eliminando las líneas que están fuera del rango de pendiente determinado y ubicación, además se analizarán las líneas en sus propios carriles correspondientes para ver si tienen pendiente positiva o negativa.

Una vez se obtienen las líneas correspondientes al carril. Se calculará la pendiente y la intersección de cada línea en particular, después se promediará la pendiente y las intersecciones para producir una línea, obteniendo como resultado el que se muestra en la figura 3.6.

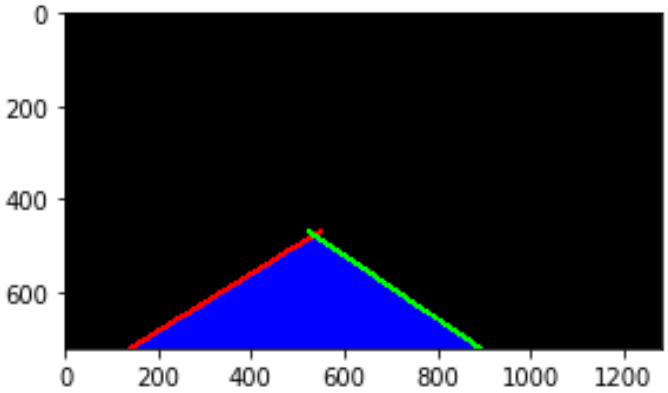


Figura 3.6 Líneas de carril

3.2.1.6 Finalmente

Tras obtener la posición en la que se encuentran las líneas, se sobreponen en la imagen original, obteniendo el resultado que se muestra en la figura 3.7.

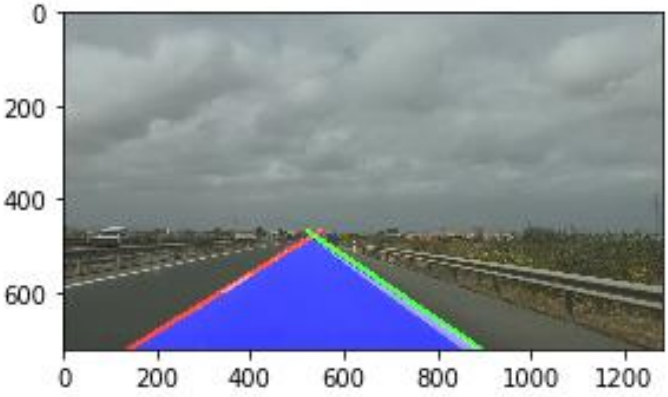


Figura 3.7 Detección de las líneas viales

3.2.2 Detección de líneas curvas

El algoritmo anterior es sólo uno de las muchos que pueden ser usados para la detección de líneas viales, sin embargo, es una forma eficaz siendo fácilmente adaptable a diferentes escalas de imagen, pero en contra para obtener un buen resultado es necesario tener buenas condiciones lumínicas como lo suele ser en cualquier proceso de detección de objetos mediante visión artificial.

En algunas ocasiones el sistema descrito anteriormente puede no ser capaz de detectar ciertos escenarios como pueden ser curvas muy pronunciadas. A continuación, se van a describir los pasos necesarios para obtener un sistema de detección de curvas usando una técnica basada en el cambio de perspectiva.

El flujograma necesario para la detección de líneas curvas se puede observar en la figura 3.8.

Figura 3.8 Flujograma de detección de líneas curvas

A continuación, se describirán los diferentes procesos se han mostrada en el flujograma.

3.2.2.1 Corrección de distorsión

Las lentes de las cámaras distorsionan la luz entrante para enfocarla en el sensor de la cámara. Aunque esto es muy útil para permitirnos capturar imágenes de nuestro entorno, a menudo terminan distorsionando la luz de forma ligeramente inexacta. Esto puede dar lugar a mediciones inexactas en aplicaciones de visión por computadora. Sin embargo, se puede corregir fácilmente esta distorsión calibrando la imagen contra un objeto conocido y generando un modelo de distorsión que tenga en cuenta las distorsiones de la lente. Este objeto es a menudo un tablero de ajedrez asimétrico.

3.2.2.2 Cambio de perspectiva

Detectar una línea curva en el espacio de una cámara puede ser complicado, pero si se pudiera ver con vista de pájaro sería más sencillo ya que se puede apreciar la forma curva en la imagen.

Esto se puede hacer realizando una transformación de cambio de perspectiva. Para ello se usa las funciones de la librería OpenCV, cv2.getPerspectiveTransform () para obtener la matriz de transformación, y cv2.warpPerspective () para aplicarla a una imagen, ambas se encuentran en las librerías de OpenCV.

En la figura 3.9 a la izquierda se muestra la imagen original y a la derecha la imagen con el cambio de perspectiva realizado.

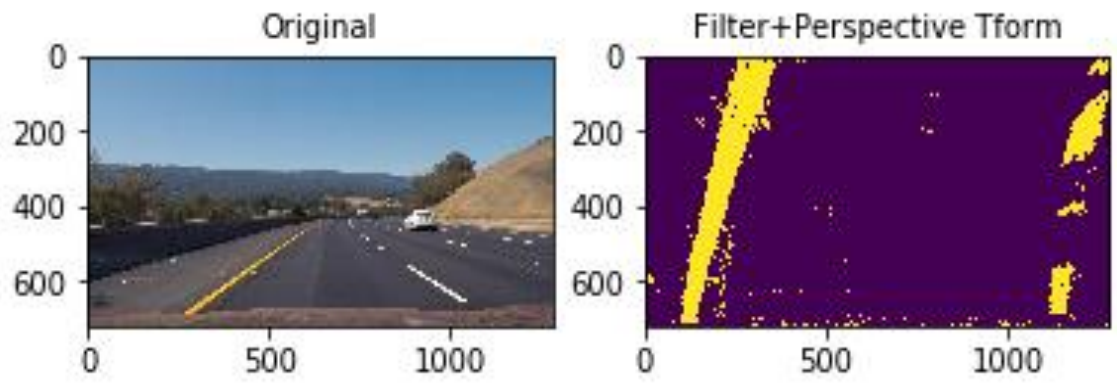


Figura 3.9 Cambio de perspectiva

3.2.2.3 Operador Sobel

Se usará un cambio en el espacio de color HLS para detectar los cambios en saturación y luminosidad. Usando el operador Sobel se obtienen los gradientes de la imagen con respecto al eje x y el y. Los pixeles que se encuentre en el umbral se extraerán formando una nueva imagen binaria que será la usada para detectar las líneas.

3.2.2.4 Detección de pico de histograma y ventana deslizante

Se recorre la imagen usando el algoritmo de ventana deslizante para detectar las líneas, estas se encontrarán en las partes con mayor número de pixeles, de esta manera si se obtiene el histograma de la imagen se tendrá las zonas donde hay mayor concentración de pixeles y se podrá partir de estas haciendo el algoritmo más eficiente.

Figura 3.10 Ejemplo histograma de imagen binaria de líneas viales

3.2.2.5 Ajuste de curvas

Se realizará el dibujo de las líneas usando la regresión polinómica, esto se hace fácilmente a través de la función np.polyfit() que se encuentra en la librería numpy, de esta manera superponiendo las líneas encontradas en la imagen original se pueden ver donde se encuentran las líneas viales.

El resultado obtenido es mostrado en la figura 3.11, a la izquierda se hallan las posiciones de las líneas, que se dibujan y finalmente se superponen en la imagen original como se muestra en el lado derecho de la figura.

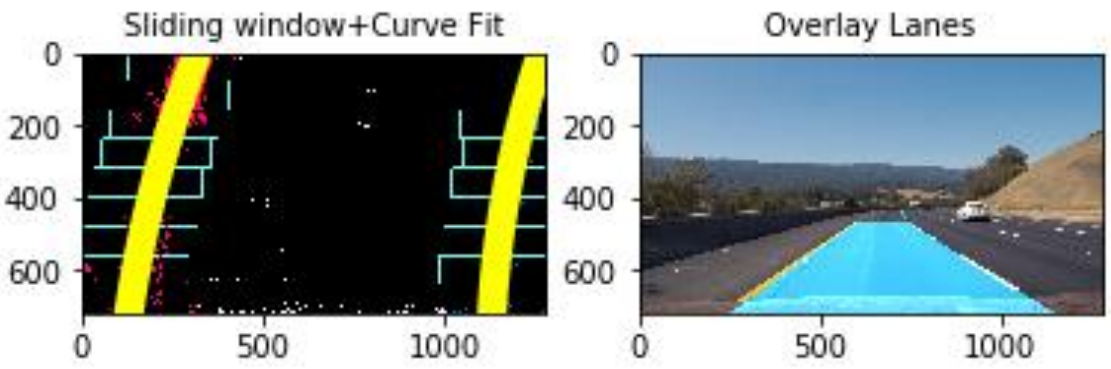


Figura 3.11 Detección de líneas curvas

3.2.3 Sistemas de detección de líneas mediante aprendizaje profundo

Con el rápido desarrollo de sensores ópticos y sensores electrónicos de alta precisión, algoritmos de visión artificial y aprendizaje automático de alta eficiencia y alta efectividad, la comprensión de la escena de conducción en tiempo real se ha vuelto cada vez más realista. Muchos grupos de investigación y la industria han invertido una gran cantidad de recursos para desarrollar algoritmos avanzados para la comprensión de la escena de conducción, apuntando a un vehículo autónomo o un sistema avanzado de asistencia al conductor (ADAS). Entre varios temas de investigación sobre la comprensión de la escena de conducción, la detección de carril es la más básica. Una vez que se obtienen las posiciones de los carriles, el vehículo sabrá a dónde ir y evitará el riesgo de toparse con otros carriles.

En este trabajo, se ha implementado un sistema de detección de carriles y se ha propuesto un método de detección de líneas viales cuando se quieren detectar carriles curvos.

Sin embargo, la mayoría de estos métodos limitan sus soluciones al detectar carriles de carretera que tienen un bajo rendimiento cuando se encuentran con escenarios de conducción desafiantes como sombras pesadas, degradación severa de marcas de carretera, oclusión grave del vehículo.

En estas situaciones, el carril puede predecirse con dirección falsa, o puede detectarse en forma parcial, o incluso no puede detectarse en absoluto. Una razón principal es que la información provista por el marco actual no es suficiente para la detección o predicción precisa del carril. En general, los carriles son estructuras lineales y continuas en el pavimento, que aparecen en forma sólida o rodada. Como las escenas de conducción son continuas y se superponen en gran medida entre dos cuadros vecinos, la posición de los carriles en los cuadros vecinos está muy relacionada. Más precisamente, el carril en el cuadro actual se puede predecir mediante el uso de múltiples cuadros anteriores, a pesar de que el carril puede sufrir daños o degradación provocada por las sombras, las manchas y la oclusión. Esto nos motiva a estudiar la detección de carriles utilizando imágenes de una escena de conducción continua.

Mientras tanto, nos damos cuenta de que el aprendizaje profundo como tecnología emergente ha demostrado un rendimiento de vanguardia, competitivo y a veces mejor que el humano en la resolución de muchos problemas de visión por computadora.

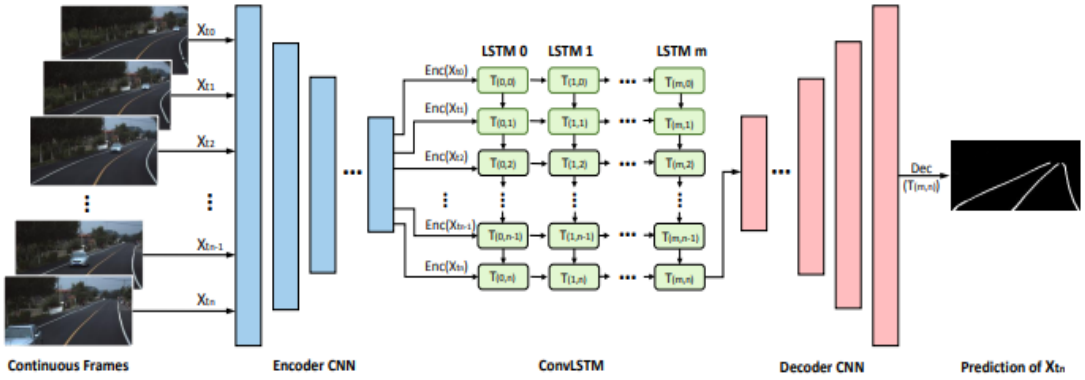


*Figura 3.12 Red neuronal propuesta en "*Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks"

Hay diferentes tipos de redes neuronales, en el artículo *"Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks*" [20] proponen una red neuronal profunda híbrida para la detección de carriles mediante el uso de múltiples imágenes de escena de conducción continua. La red neuronal profunda híbrida propuesta combina dos redes neuronales. Se puede decir que es una red neural convolucional, que toma múltiples cuadros como entrada, y predice el carril del cuadro actual.

Las dos principales ventajas de usar una red neuronal para la detección de líneas son:

- Las redes pueden ser entrenadas para detectar con precisión líneas usando una sola imagen situaciones de sombra, degradación de la carretera y oclusiones que provocan los demás vehículos que circulan por el entorno del vehículo. Como se puede extraer más información de múltiples imágenes continuas que de una imagen actual, el método puede predecir con mayor precisión el carril en comparación con los métodos basados en una sola imagen, especialmente en el manejo de las situaciones desafiantes mencionadas anteriormente.
- Pueden ser entrenadas con multitud de dataset en diferentes condiciones y diferentes tipos de carretera por lo que no sería necesario adaptar el algoritmo a cada condición.

También existen desventajas:

- El número de muestras necesario para entrenar las redes puede ser muy grande por lo que hacerse con este tipo de dataset puede ser una tarea complicada y tediosa.
- El coste de los ordenadores necesarios para el entrenamiento de las redes neuronales y su posterior aplicación.
- El tiempo de ejecución de las redes neuronales es mayor que la de los algoritmos tradicionales de visión por computadora en algunas ocasiones.

CAPÍTULO 4

SISTEMA DE DETECCIÓN DE SEÑALES DE TRÁFICO

4.1 Introducción

El reconocimiento de señales de tráfico es una tecnología que llevan incorporada la nueva generación de coches de tal manera que son capaces de identificar estas señales durante la circulación. Este es un problema del mundo real de gran relevancia industrial. Aunque los sistemas comerciales han llegado al mercado y se han publicado varios estudios sobre este tema, el reconocimiento de señales de tráfico es un problema de clasificación multiclase. Las señales de tráfico pueden proporcionar una amplia gama de variaciones entre clases en términos de color, forma y la presencia de pictogramas o texto. Sin embargo, existen subconjuntos de clases (por ejemplo, señales de límite de velocidad) que son muy similares entre sí. El clasificador tiene que hacer frente a grandes variaciones en las apariencias visuales debido a cambios de iluminación, oclusiones parciales, rotaciones, condiciones climáticas, etc. Los humanos son capaces de reconocer la gran variedad de señales de tráfico existentes con una exactitud cercana al 100%. Esto no solo se aplica a la conducción en el mundo real, que proporciona vistas tanto contextuales como múltiples de una sola señal de tráfico, sino también el reconocimiento de imágenes individuales.

En este capítulo se expone cómo funcionan las CNN (Redes Neuronales Convolucionales), se realiza el desarrollo un sistema de detección de señales de tráfico mediante una CNN y se comprobara su rendimiento.

4.2 Obtención de datos

Una de las partes más complicadas de obtener a la hora de realizar un proyecto con redes neurales es la gran cantidad de datos necesarios para su entrenamiento. Para poder tener una buena eficiencia es necesario tener un dataset muy grande, esto es una tarea que podría llevar muchas horas, pero por suerte se pueden encontrar sitios web en internet donde han recopilado datos y los ponen a disposición de quien los desee para su estudio.

Para el entrenamiento de la red neuronal convolucional se ha utilizado un dataset de señales alemanas que se puede encontrar en [21].

Las características más relevantes son:

- Contiene 43 clases diferentes de señales.
- Consta de 50000 imágenes en total.
- Una base de datos grande y realista.
- El tamaño de las imágenes varía entre 15x15 y 250x250 pixeles.
- Las imágenes no son necesariamente cuadradas.
- La señal de tráfico real no está necesariamente centrada dentro de la imagen.
- Cada imagen contiene una señal de tráfico.

Figura 4.1 Ejemplo de imágenes de señales del dataset

4.3 Extracción de los datos

En primer lugar, hay que redimensionar todas las imágenes para que tengan el mismo tamaño, en este proyecto las imágenes se han redimensionado a 32x32x3 pixeles.

El dataset se ha dividido en tres grupos. A continuación, se describe para el uso que se le ha dado a cada conjunto.

1. Un conjunto de datos que se usará para el aprendizaje de la red neuronal, al cual se le llamará "ejemplos de entrenamiento" permitiendo obtener los pesos.
2. Un conjunto de datos que se usará para ajustar los parámetros de la red neuronal, al cual se le llamará "ejemplos de validación".
3. Un conjunto de datos que se usará solo para evaluar el rendimiento de la red neuronal, al cual se le llamará "ejemplos de prueba", estos datos nunca se usarán para el entrenamiento consiguiéndose así una prueba de rendimiento con imágenes que la red no conoce.

Finalmente, el dataset queda como se puede ver en la tabla 4.1.

Número de ejemplos de entrenamiento (69,5%)	34799
Número de ejemplos de validación (25,2%)	12630
Número de ejemplos de prueba (8,8%)	4410
Tamaño de las imágenes (pixeles)	32x32x3
Número de clases	43

Tabla 4.1 Tamaño de los conjuntos de datos

En la figura 4.2 se muestran imágenes del dataset, una vez que se han redimensionado al tamaño de 32x32x3 pixeles.

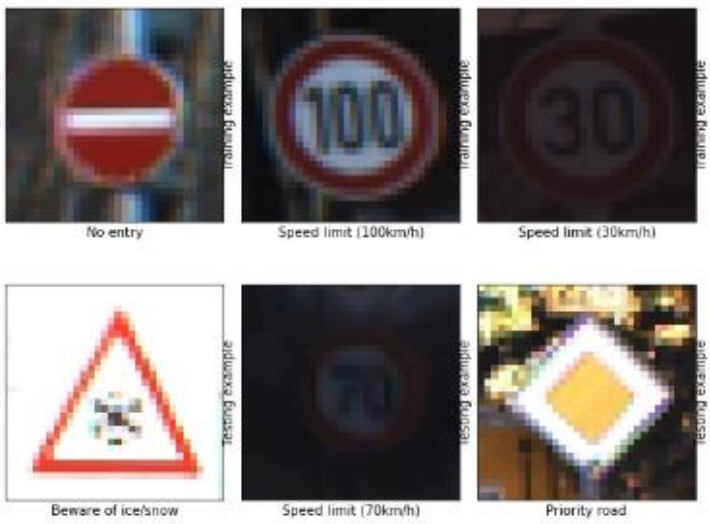


Figura 4.2 Ejemplos de imágenes de entrenamiento

De las diferentes clases de señales del dataset, hay un número diferente para cada clase.

En las figuras 4.3, 4.4 y 4.5 se ha representado un histograma con el número de señales que se tiene de cada clase en los diferentes conjuntos de entrenamiento.

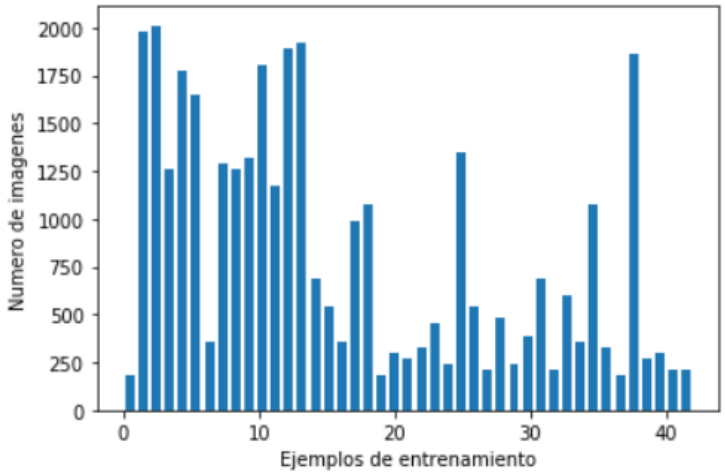


Figura 4.3 Histograma del número de ejemplos de entrenamiento

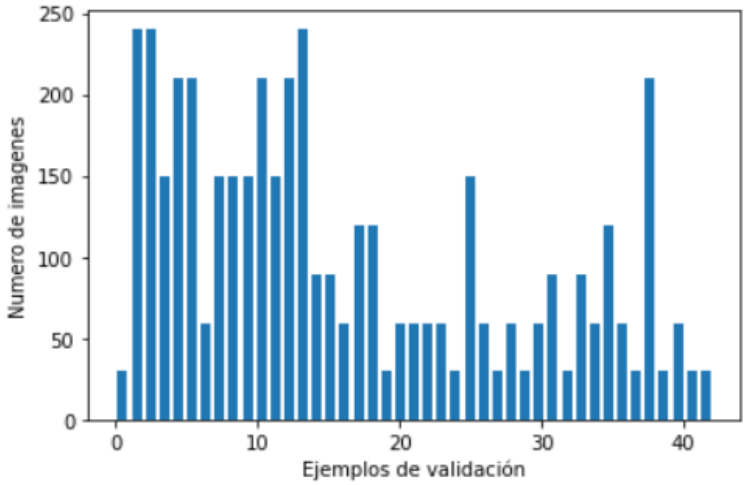


Figura 4.4 Histograma del número de ejemplos de validación

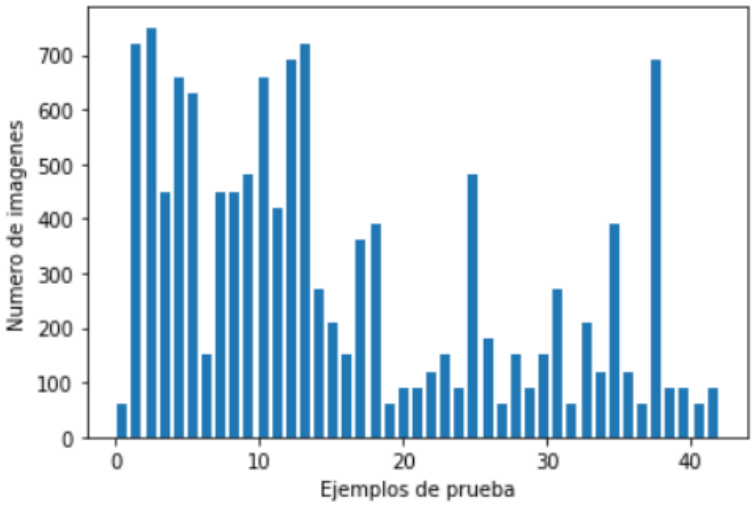


Figura 4.5 Histograma del número de ejemplos de prueba

Las diferentes clases de señales que se encuentran en el dataset se muestran en la tabla 4.2.

Clase	Nombre de la señal	Clase	Nombre de la señal
0	Speed limit (20km/h)	**22**	Bumpy road
1	Speed limit (30km/h)	**23**	Slippery road
2	Speed limit (50km/h)	**24**	Road narrows on the right
3	Speed limit (60km/h)	**25**	Road work
4	Speed limit (70km/h)	**26**	Traffic signals
5	Speed limit (80km/h)	**27**	Pedestrians
6	End of speed limit (80km/h)	**28**	Children crossing
7	Speed limit (100km/h)	**29**	Bicycles crossing
8	Speed limit (120km/h)	**30**	Beware of ice/snow
9	No passing	**31**	Wild animals crossing
10	No passing for vehicles over 3.5 metric tons	**32**	End of all speed and passing limits
11	Right-of-way at the next intersection	**33**	Turn right ahead
12	Priority road	**34**	Turn left ahead
13	Yield	**35**	Ahead only
14	Stop	**36**	Go straight or right
15	No vehicles	**37**	Go straight or left
16	Vehicles over 3.5 metric tons prohibited	**38**	Keep right
17	No entry	**39**	Keep left
18	General caution	**40**	Roundabout mandatory
19	Dangerous curve to the left	**41**	End of no passing
20	Dangerous curve to the right	**42**	End of no passing by vehicles over 3.5 metric tons
21	Double curve		

Tabla 4.2 Clases de señales de tráfico

4.4 Preprocesamiento de los datos

Para obtener un resultado óptimo se llevará a cabo un preprocesamiento de las imágenes antes de ser usadas en la red neuronal.

Los procesos que se han llevado a cabo son:

1. Randomización los datos de entrenamiento.

Se trata de conseguir que el conjunto de datos de entrenamiento sea más aleatorio y así aumentar la variedad en el conjunto de datos, lo cual consigue que el modelo sea más estable.

2. Convertir en escala de grises.

Se cambiaran las imágenes a escala de grises, que como se puedes ver en la publicación de LeCun [22], en la que se demostró que el uso de imágenes en escala de grises en lugar de color mejora el rendimiento en redes neuronales convolucionales.

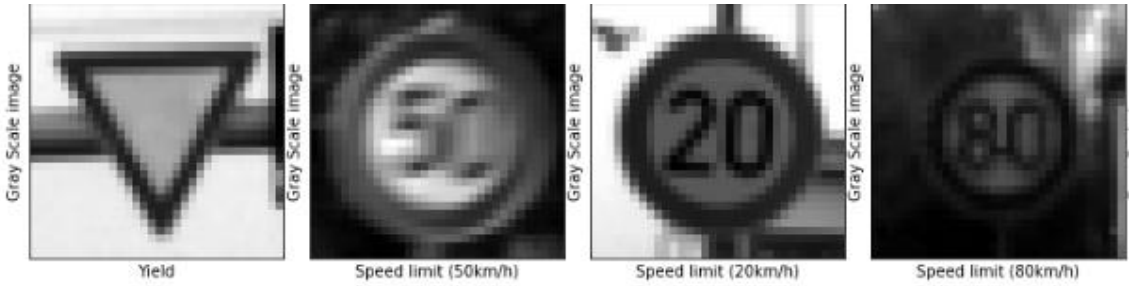


Figura 4.6 Imágenes en escala de grises del dataset

3. Ecualización de histograma local.

Esta técnica simplemente extiende los valores de intensidad más frecuentes en una imagen, lo que da como resultado una mejora de las imágenes que tienen bajo contrate. La aplicación de esta técnica es muy útil debido a que el conjunto de datos usado contiene imágenes tomadas en condiciones reales lo que hace que muchas de ellas tengan bajo contraste.

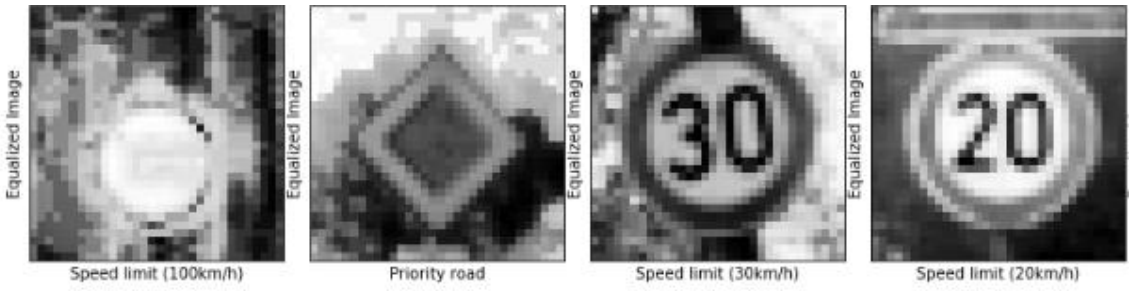


Figura 4.7 Imágenes en las que se ha aplicado la ecualización del histograma local del dataset

4. Normalizar.

La normalización es un proceso que cambia el rango de valores de intensidad de píxeles. Se divide el valor de todos los pixeles entre 255, de esta manera el valor de gris más pequeño es llevado a 0 y el máximo 1 Por lo general, los datos de las imágenes deben normalizarse para

tener una cierta independencia de las propiedades de la imagen, como lo son el brillo y el contraste.

4.5 Redes neuronales convolucionales

En este apartado, se diseñará e implementará un modelo de aprendizaje profundo que se usará para reconocer las señales de tráfico del dataset.

Cuando se trabaja con imágenes lo normal es usar redes neuronales convolucionales (CNN), estas pueden aprender a relacionar una entrada con una salida donde la entrada es la imagen. El funcionamiento básico de la red son operaciones de convolución y normalmente son usadas para tareas de:

- Detección/categorización de objetos
- Clasificación de escenas
- Clasificación de imágenes en general

Hoy en día el tamaño estándar de las imágenes es de 1980x1080 pixeles, (como pueden ser las imágenes tomadas por la mayoría de móviles) pero esto podría ser un problema a la hora de entrenar las CNN ya que son redes full-connected, es decir, cada una de las neuronas que compone la primera capa oculta se conecta con todos y cada uno de los pixeles de la imagen de entrada a la vez. Por ejemplo, si se tiene una imagen de un tamaño de 96 x 96 pixeles se obtienen alrededor de 10^4 elementos de entrada y, si se asume que se quieren aprender 100 características, el número de elementos a entrenar sería del orden de 10^6. Esta enorme cantidad de elementos que deben ser entrenados hace que la duración del entrenamiento sea extraordinariamente larga, dificultando, en gran medida, el uso de las redes neuronales convencionales para aplicaciones con imágenes, por ello suelen usarse tamaños de imágenes pequeños. Para este proyecto se usará un tamaño de 32x32pixeles que es un tamaño asumible para que el sistema pueda funcionar correctamente sin ser demasiado lento, ya que al utilizar mayores resoluciones los tiempos de entrenamiento y testeo se vuelven enormes.

Una red neuronal convolucional [23] es un tipo de red multicapa que consta de diversas capas convolucionales, capas de pooling (submuestreo) alternadas, junto con una serie de capas full-connected. La entrada de una capa convolucional suele ser, generalmente, una imagen *m x n x r*, donde m es la altura, n es el ancho de la imagen y r es el número de canales.

En la figura 4.8 se muestra un ejemplo de una CNN, paso por paso.

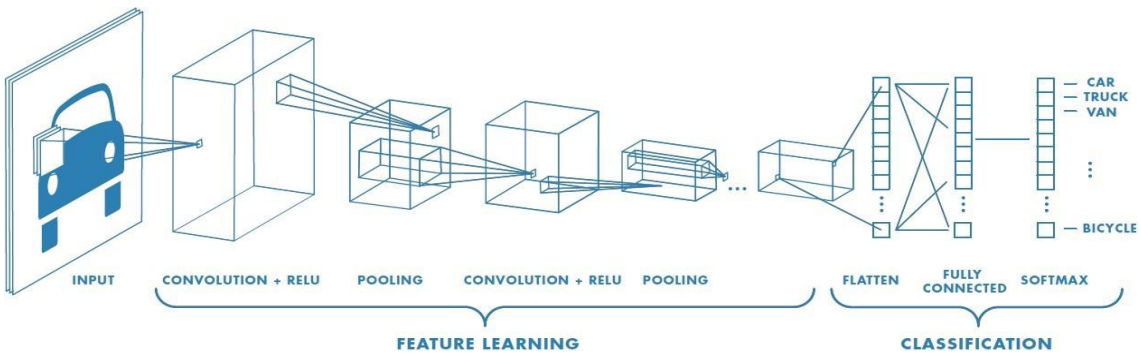


Figura 4.8 Red Neuronal Convolucional

A continuación, se expondrá con más profundidad en que consiste las diferentes capas y procesos que se llevan a cabo en una CNN [24].

- Convolutional Layer (Capa convolucional)

La capa convolucional es el componente básico de una red neuronal convolucional. En ella se realiza la mayor parte del trabajo computacional, mediante el uso de una operación llamada convolución, la cual recibe como entrada o input la imagen y luego aplica sobre ella un filtro o kernel que nos devuelve un mapa de las características de la imagen original. La convolución aprovecha tres ideas importantes a remarcar:

- Interacciones dispersas, ya que al aplicar un filtro de menor tamaño sobre la entrada original podemos reducir drásticamente la cantidad de parámetros y cálculos.
- Los parámetros compartidos, que hace referencia a compartir los parámetros entre los distintos tipos de filtros, ayudando también a mejorar la eficiencia del sistema.
- Las representaciones equivalentes, las cuales indican que si las entradas cambian, las salidas van a cambiar también en forma similar.

Supongamos que la matriz de entrada es $X= \{x_{i,j} \mid i=1,2,..I, j=1,2,...J\}$, siendo I=32 y J=32. El núcleo de convolución se denota como $W=\{w_{m,n} \mid m=0,1,..F-1, n=0,1,...F-1\}$, donde F denota el tamaño del núcleo convolucional, que son iguales. En la Figura 4.9, F es igual a 5. La expresión de la capa convolucional se muestra en la ecuación (17).

$$a_{i,j} = \left\{ f\left(\sum_{m=0}^{F-1} \sum_{n=0}^{F-1} w_{m,n} x_{i+m,j+n} + b \right) \right\}_{i=1,2...I;j=1,2...,J} \tag{17}$$

Donde, $a_{i,j}$ representa la salida después de la convolución, b denota el término de compensación para cada convolución y f() denota la función de activación.

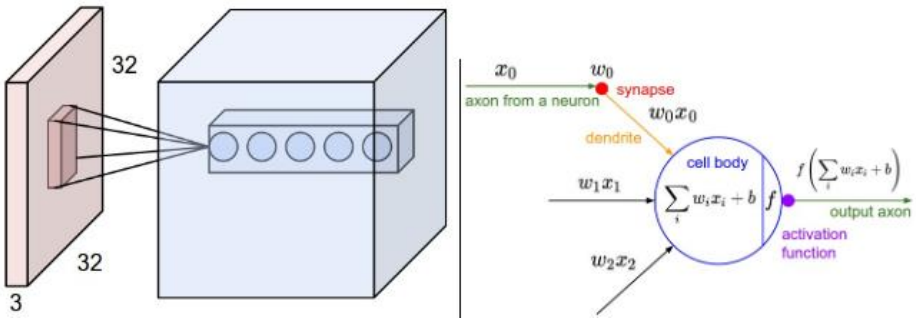


Figura 4.9 Izquierda: Ejemplo de una entrada de una imagen de 32x32x3 pixeles con través de 5 neuronas conectadas. Derecha: Cálculos que se producen dentro de la neurona para obtener los pesos

- Activation Functions (Función de activación)

En general, hay cinco funciones de activación ampliamente utilizadas, que son Sigmoid, Tanh, ReLU, Softplus y Gaussian [21]. Sigmoide, Tanh y Gaussian son generalmente funciones no lineales saturantes, que se muestran en las ecuaciones (18), (19) y (20) respectivamente. Estas se eligen principalmente como funciones de activación en las CNN tradicionales.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{18}$$

$$f(x) = \frac{1}{e^{x} + e^{-x}} \tag{19}$$

$$f(x) = e^{-x^2} \tag{20}$$

Actualmente, las funciones no lineales no saturadas a menudo se usan como funciones de activación en estructuras CNN. Las funciones más utilizadas son las funciones ReLU y Softplus, que se muestran en las ecuaciones (21) y (22), respectivamente.

$$f(x) = \max(0, x) \tag{21}$$

$$f(x) = \ln(1 + e^{x}) \tag{22}$$

Las cinco funciones de activación se muestran en la Figura 4.10, donde se puede observar que el espacio de salida de la función Sigmoide y Gaussiana está en (0, 1), y el espacio de salida de la función Tanh está en (−1, 1). Cuando la entrada es demasiado grande, la salida de la función Sigmoide y la función Tanh tiende a 1 y permanece estable, pero la función gaussiana tiende a 0 a medida que aumenta la entrada.

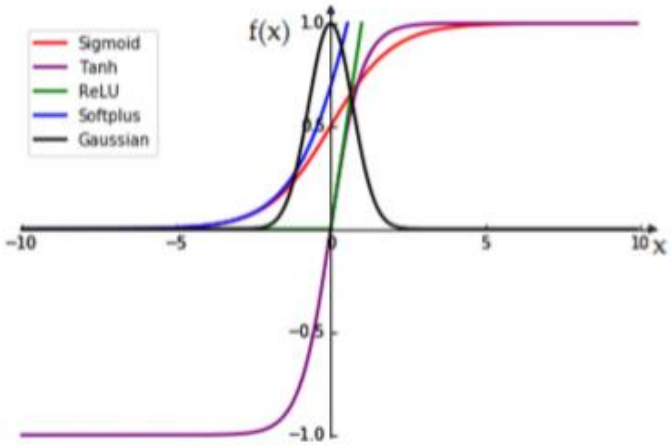


Figura 4.10 Funciones de activación

Cuando la entrada es demasiado pequeña, la salida de la función Sigmoide y la función Gaussiana tiende a 0 y permanece estable, pero la función Tanh tiende a -1 a medida que disminuye la entrada. La salida de las tres funciones de activación puede estar cerca de ser suave. Por lo tanto, su gradiente es muy cercano a cero, lo que no es propicio para actualizar el peso. Del análisis anterior, se puede concluir que existe un problema de explosión de gradiente y desaparición de gradiente en funciones saturadas no lineales. Se puede ver en las líneas verde y negra de la Figura 4.9 que la función ReLU y la función gaussiana no tienen problemas de saturación de gradiente cuando la entrada es positiva, y son mucho más rápidas que las funciones no lineales saturadas. En la referencia [25], los autores demuestran que la función ReLU es la que mejor funciona para modelos de CNN, por lo que se ha elegido para el entrenamiento de los modelos que se usan en el proyecto.

- Pooling layer (Neuronas de Reducción de Muestreo)

Es común insertar periódicamente una capa de agrupación entre capas sucesivas de convolución en una arquitectura CNN. El propósito de la capa de agrupación es realizar un proceso de selección de características para reducir las dimensiones de los datos mientras se conservan las características principales de los datos. La agrupación máxima, la agrupación media y la agrupación aleatoria son enfoques generalmente utilizados, que extraen los puntos con el mayor valor, valor medio y valores aleatorios en el dominio local aceptado. La expresión de la capa de agrupación se muestra en la ecuación (23), donde pool () representa la operación de agrupación máxima (poolmax), en la figura 4.11 se muestra el funcionamiento de esta función. Generalmente, la salida de la primera capa se denota como a_n^l y a_n^{l-1} se denota como la salida de la capa anterior, donde n corresponde a la enésima muestra en la ecuación (23).

$$a_n^l = pool(a_n^{l-1}) \tag{23}$$

En las CNNs, la combinación de convolución, ReLU y agrupación desempeña el papel de extracción de características.

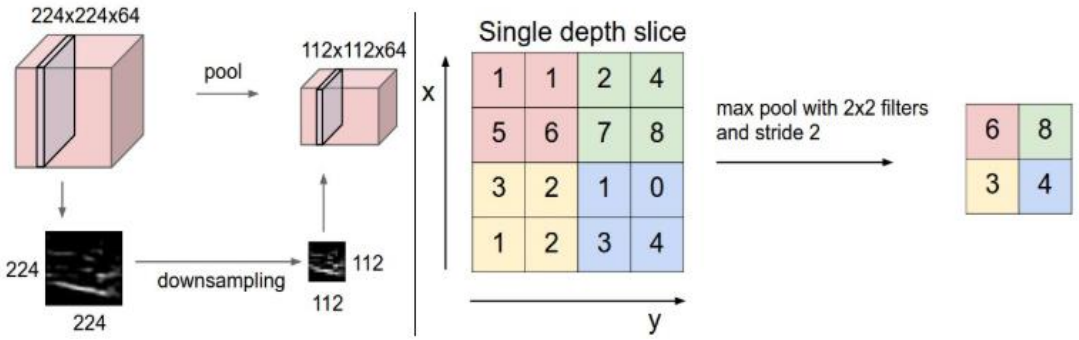


Figura 4.11 Izquierda: La capa de agrupación reduce el volumen espacialmente, independientemente en cada segmento de profundidad del volumen de entrada. Izquierda: en este ejemplo, el volumen de entrada de tamaño [224x224x64] se agrupa con el tamaño de filtro 2, paso 2 en el volumen de salida de tamaño [112x112x64]. Tenga en cuenta que la profundidad del volumen se conserva. Derecha: La operación de disminución de resolución más común es max, dando lugar a la agrupación máxima, que se muestra aquí con un paso de 2. Es decir, cada máximo se toma sobre 4 números (pequeño cuadrado de 2x2).

- Fully Connected Layer (Capa totalmente conectada)

La capa totalmente conectada es generalmente la última capa en la estructura de CNN. Cada neurona usa la función de activación ReLU, que está completamente vinculada a las neuronas de la capa anterior. La capa totalmente conectada puede integrar información local, que tiene la capacidad de discriminar clases y la salida de neuronas se pasa a la capa de salida. Por lo tanto, la capa completamente conectada tiene algún papel de clasificadores convencionales. Si la capa h es la capa completamente conectada, la salida de esta capa estará compuesta por la ecuación (24), donde w^l denota el núcleo convolucional y b^l denota el término de desplazamiento:

$$a_n^l = f\left(w^l \cdot a_n^{l-1} + b^l\right) \tag{24}$$

- Output Layer (Capa de salida)

La capa de salida también llamada capa softmax, que está representada por la ecuación (25). La función softmax se usa principalmente en el proceso de clasificación múltiple, que asigna la salida de la capa completamente conectada a (0, 1). Cada salida corresponde a la probabilidad de clasificación, y su suma acumulativa es 1. Finalmente, la clasificación de la probabilidad máxima se selecciona como salida. El proceso de la función softmax se muestra en la figura 4.12.

$$a_n^l = o_n^v = softmax(w^L \cdot a_n^{L-1} + b^L) \tag{25}$$

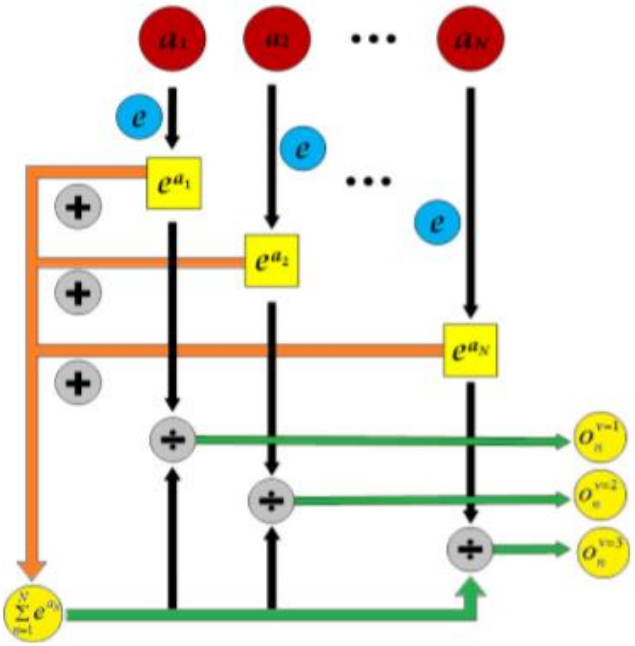


Figura 4.12 Proceso de clasificación softmax

La probabilidad de diferentes categorías de clasificación obtenidas por softmax se denota por $o_x^v(v = 1,2,3,\ldots V; n = 1,2,3,..N)$, que indica la probabilidad de salida de la enésima muestra para v diferentes categorías clasificadas. Si t_n^v representa la probabilidad de salida esperada de la enésima muestra en *v* diferentes categorías de clasificación, la fórmula de error E_n correspondiente a la enésima muestra se obtendrá mediante la ecuación (26).

$$E_n = \frac{1}{2}\sum_{v=1}^{V} \|t_n^v - o_n^v\|_2^2 \tag{26}$$

y el error global de N muestras podría obtenerse mediante la ecuación (27).

$$E = \sum_{n=1}^{V} E_n \tag{27}$$

Según los análisis anteriores, la capa totalmente conectada y la capa de salida podrían ser equivalentes. Cuando la red CNN se entrena con datos pequeños, los resultados del

entrenamiento son propensos a un sobreajuste. Para evitar el sobreajuste, la técnica de abandono evita que algunas neuronas aleatorias realicen la propagación de CNN. Por lo tanto, el aprendizaje de las neuronas tiene características más robustas. En la actualidad, la mayoría de las investigaciones de CNN adopta ReLU y la tecnología de abandono, que ha logrado un buen rendimiento de clasificación.

4.6 Arquitecturas de las CNN

Existen diversas arquitecturas de CNN, en este apartado se han recopilado algunas de las redes convolucionales más utilizadas por investigadores y grandes empresas como Google, Amazon, o IBM. A continuación, se exponen algunas de estas redes:

- **LeNet:** Las primeras aplicaciones exitosas de Redes Convolucionales fueron desarrollada por Yann LeCun [26] en la década de 1990. De estos, el más conocido es la arquitectura LeNet que se utilizó para leer códigos postales, dígitos, etc.

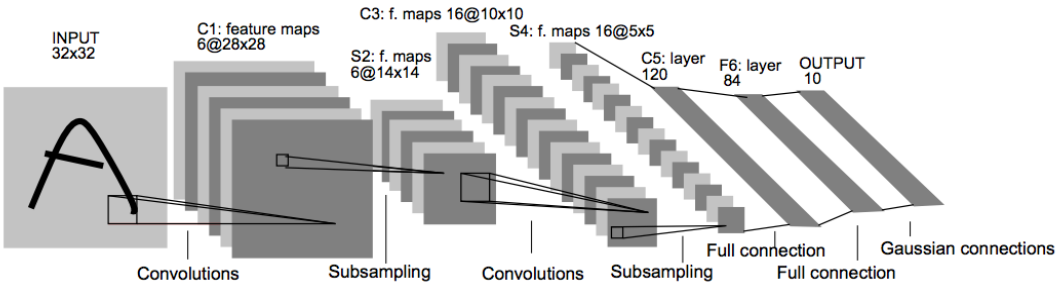


Figura 4.13 Estructura LeNet

- **AlexNet:** El primer trabajo que popularizó las redes convolucionales en visión artificial fue AlexNet [27], desarrollado por Alex Krizhevsky, Ilya Sutskever y Geoff Hinton. El AlexNet se presentó al desafío ImageNet ILSVRC en 2012 y superó significativamente al segundo finalista (error de los 5 primeros del 16% en comparación con el segundo con el 26% de error). La red tenía una arquitectura muy similar a LeNet, pero era más profunda, más grande y presentaba capas convolucionales apiladas una encima de la otra (anteriormente era común tener una sola capa convolucional siempre seguida inmediatamente por una capa pool).

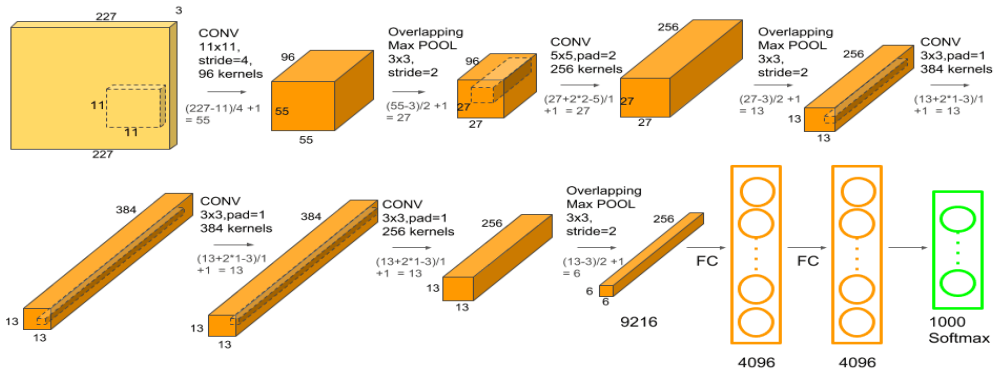


Figura 4.14 Estructura AlexNet

- **ZF Net**: El ganador del ILSVRC 2013 fue una Red Convolucional de Matthew Zeiler y Rob Fergus[28]. Se hizo conocido como ZFNet (abreviatura de Zeiler & Fergus Net). Fue una mejora en AlexNet al ajustar los hiperparámetros de la arquitectura, en particular al expandir el tamaño de las capas convolucionales medias y reducir el tamaño del paso y el filtro en la primera capa.
- **GoogLeNet**. El ganador de ILSVRC 2014 fue una Red Convolucional de Szegedy[29] et al. de Google. Su principal contribución fue el desarrollo de un módulo de inicio que redujo drásticamente el número de parámetros en la red (4M, en comparación con AlexNet con 60M). Además, este documento utiliza una agrupación promedio en lugar de capas Completamente conectadas en la parte superior de ConvNet, eliminando una gran cantidad de parámetros que no parecen importar mucho. También hay varias versiones de seguimiento de GoogLeNet, la más reciente Inception-v4.
- **VGGNet:** El segundo lugar en ILSVRC 2014 fue la red de Karen Simonyan y Andrew Zisserman que se conoció como VGGNet [30], su arquitectura se muestra en la figura 4.15. Su principal contribución fue demostrar que la profundidad de la red es un componente crítico para un buen rendimiento. Su mejor red final contiene 16 capas convolucionales y de manera atractiva, presenta una arquitectura extremadamente homogénea que solo realiza convoluciones 3x3 y agrupación 2x2 desde el principio hasta el final. Su modelo previamente entrenado está disponible para su uso. Una desventaja de VGGNet es que es más costoso de evaluar y utiliza mucha más memoria y parámetros (140M). La mayoría de estos parámetros están en la primera capa completamente conectada, y desde entonces se descubrió que estas capas se pueden eliminar sin rebajar el rendimiento, lo que reduce significativamente la cantidad de parámetros necesarios.

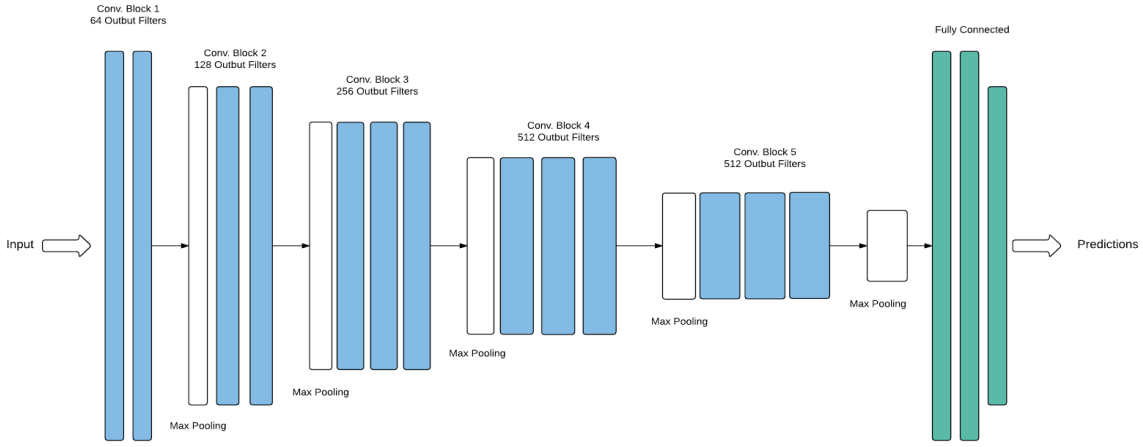


Figura 4.15 Estructura VGGNet

- **ResNet**: Residual Network [31] desarrollada por Kaiming He et al. fue el ganador de ILSVRC 2015. Cuenta con conexiones especiales de omisión y un uso intensivo de la normalización por lotes [32]. ResNets son actualmente modelos de redes neuronales convolucionales de última generación y son la opción predeterminada para usar CNN en la práctica. Hoy en día ya se pueden ver desarrollos más recientes que modifican la arquitectura original de Kaiming He et al [33].

En los próximos apartados, se presentarán con más detalle las arquitecturas LeNet y VGGNet ya que estas han sido adaptadas para la implementación del detector de señales de tráfico.

4.6.1 LeNet

LeNet es una red convolucional diseñada para el reconocimiento de caracteres escritos a mano e impresos en máquina. Aunque esta CNN está diseñada para clasificar dígitos escritos a mano, se tendrá una precisión muy alta cuando se trata de señales de tráfico, dado que tanto los dígitos escritos a mano como las señales de tráfico se entregan a la computadora en forma de imágenes de píxeles.

Esta red convolucional consta de 5 capas, en las cuales se han realizado los procesos que son descritos en la figura 4.16.

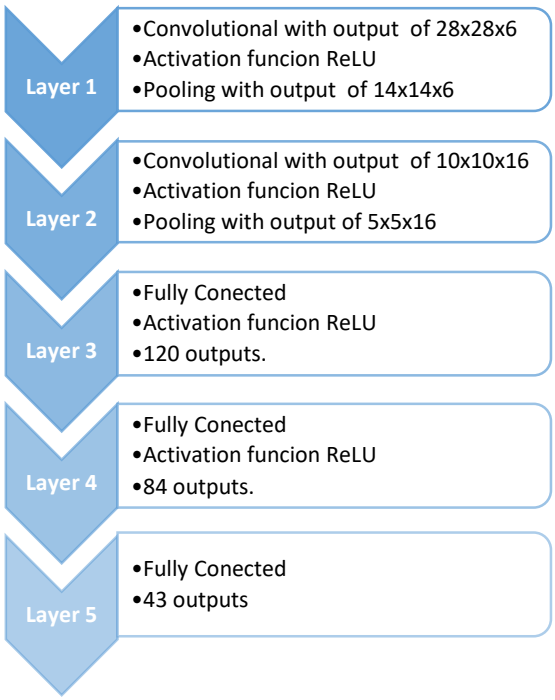


Figura 4.16 Estructura LeNet para detectar señales de tráfico

4.6.2 VGGNet

La red VGGNet fue diseñada mientras se investigaba la profundidad de la red convolucional para el reconocimiento de imágenes a gran escala. Su principal contribución es una evaluación exhaustiva de redes de profundidad creciente utilizando una arquitectura con filtros de convolución muy pequeños (3x3), lo que demuestra que se puede lograr una mejora significativa en las configuraciones de la técnica anterior empujando la profundidad a 16-19 capas de peso. Por su buen rendimiento, se ha seleccionada para adaptarse detectando imágenes que contienen señales de tráfico.

La arquitectura original de VGGNet tiene 16-19 capas, pero se han excluido algunas de ellas y se implementara una versión modificada de solo 12 capas para ahorrar recursos computacionales.

Los procesos llevados a cabo en cada capa son descritos en la figura 4.17.

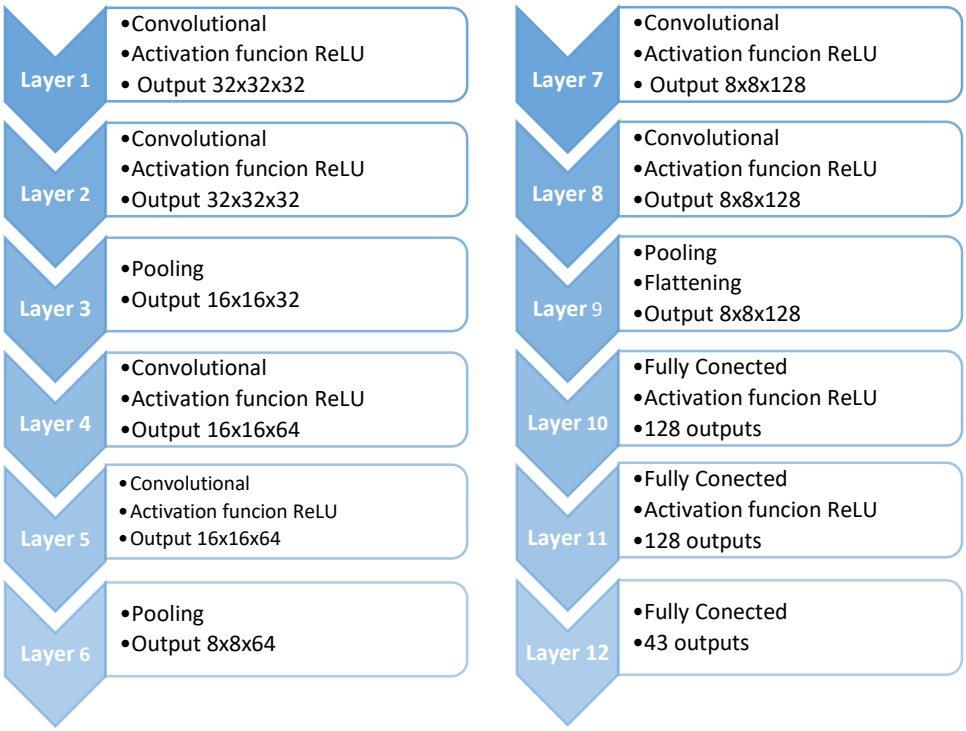


Figura 4.17 Estructura VGGNet para detectar señales de tráfico

4.7 Modelo de entrenamiento y evaluación

En esta sección del proyecto, se mostrará como se ha procedido para entrenar las redes LeNet y VGGNet con el conjunto de imágenes de entrenamiento y validación. Para el entrenamiento se ha usado la versión 2 de TensorFlow [34] escrita para el lenguaje de programación Python. TensorFlow es una biblioteca de código abierto desarrollada por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje.

Para entender algunos de los procesos que se van a llevar a cabo en el entrenamiento del modelo, se van a definir los siguientes términos.

- **Batch size:** El número de ejemplos de entrenamiento en una pasada hacia adelante y hacia atrás. Cuanto mayor sea el tamaño del lote (un conjunto de muestras), más espacio de memoria necesitará. Por ejemplo, si se supone que se tiene un lote de 1050 muestras de entrenamiento y se desea configurar batch size igual a 100. El algoritmo toma las primeras 100 muestras (del 1 al 100) del conjunto de datos de entrenamiento. A continuación, toma las segundas 100 muestras (de la 101 a la 200) y vuelve a formar la red.
- **Epoch:** una "época" describe la cantidad de veces que el algoritmo ve TODO el conjunto de datos. Entonces, cada vez que el algoritmo ha visto todas las muestras en el conjunto de datos, se completa una época.
- **Iteración:** describe la cantidad de veces que un "lote" de datos pasó a través del algoritmo.

Para entender mejor estos conceptos, se expone un ejemplo numérico:

"Si se supone que se tiene un conjunto de datos de 10 ejemplos / muestras. Tiene un bach size de 2 y se ha especificado que el algoritmo se ejecute durante 3 epochs. Se tendría en cada época 5 batch (10/2 = 5). Cada batch se pasa a través del algoritmo, por lo tanto, tiene 5 iteraciones por época. Como se ha especificado 3 épocas, tiene un total de 15 iteraciones (5 * 3 = 15) para el entrenamiento."

El entrenamiento de la red se hará de la siguiente forma:

- Antes de cada epoch, se randomizará el conjunto de entrenamiento.
- Se entrenará el modelo con el conjunto de datos de entrenamiento que han sido previamente preprocesados.
- Se usará el dropout, es probable que las redes neuronales profundas sobren ajusten un conjunto de datos de entrenamiento. Esto provoca que el modelo aprenda el ruido de los datos de entrenamiento, produciendo un bajo rendimiento cuando se evalúan nuevos datos. El error aumenta debido al sobreajuste. El dropout permite que, durante el entrenamiento, se ignoren al azar un porcentaje de las unidades o neuronas de entrada. Esto da lugar a una red neuronal más pequeña que la original y por tanto con menos parámetros, lo que disminuye la dependencia entre neuronas y evita así el sobre entrenamiento.
- Para saber cómo de alejadas están las predicciones de un modelo de sus etiquetas reales se utiliza la función de perdida, en este caso se ha usado la función Cross-Entropy, que es especialmente útil en problemas donde hay múltiples clases excluyentes. La función viene dada por la ecuación (28).

$$H(y,\hat{y}) = \sum_{x} y_x \log \frac{1}{\hat{y}_x} = \sum_{x} y_x \log \hat{y} \tag{28}$$

  Donde x es una variable discreta e $\hat{y}$ la predicción para la distribución real y.
- Se minimiza la función de pérdida usando el Algoritmo de Adaptative Moment Estimation (Adam), es una función que permite modificar el ritmo de aprendizaje (o learning rate) a medida que se desarrolla el entrenamiento. En concreto se trata de una variante del descenso de gradiente estocástico o stochastic gradient descent. La red neuronal contiene una serie de hiperparametros como la tasa de aprendizaje, si su valor es demasiado pequeño la convergencia y por lo tanto el aprendizaje será demasiado lento, en cambio sí es demasiado grande el siguiente punto se excederá del mínimo por lo que un valor adecuado del learning rate es clave. El descenso de gradiente es un proceso iterativo donde los pesos de la red se actualizan en cada iteración para disminuir la desviación de la salida.
- Se especifica la tasa de aprendizaje a 0,001, esto ayuda a controlar el ajuste de los pesos del modelo con respecto al descenso de gradiente.
- Se obtiene la entropía cruzada de softmax entre predicciones y etiquetas.
- Después de cada epoch, se medirá la pérdida y la precisión del conjunto de validación.

Al finalizar el entrenamiento, si se obtiene una baja precisión en los conjuntos de entrenamiento y validación implicará un ajuste insuficiente, mientras que alta precisión en el conjunto de entrenamiento, pero una baja precisión en el conjunto de validación implicará un sobreajuste.

Para el entrenamiento se define un batch size de 64 sobre un total de 34799 imágenes del conjunto de entrenamiento, lo que provoca un total de 644 iteraciones, se han establecido 30 epochs o recorridos sobre el dataset completo y un learning rate de 0,001.

Una vez definido todo el proceso, se lleva a cabo el entrenamiento de ambas redes mediante la librería TensorFlow. En la tabla 4.3 se muestran los parámetros usados para el entrenamiento y los resultados del conjunto de validación después de cada epoch.

Configuración para el entrenamiento y resultados de las redes LeNet y VGGNet		
EPOCHS	**BATCH SIZE**	**LEARNING RATE**
30	64	0,001

Precisión de validación red LeNeT	**Precisión de validación red VGGNet**
EPOCH 1 : Validation Accuracy = 78.186%	EPOCH 1 : Validation Accuracy = 15.873%
EPOCH 2 : Validation Accuracy = 86.417%	EPOCH 2 : Validation Accuracy = 43.560%
EPOCH 3 : Validation Accuracy = 88.912%	EPOCH 3 : Validation Accuracy = 73.265%
EPOCH 4 : Validation Accuracy = 90.295%	EPOCH 4 : Validation Accuracy = 85.442%
EPOCH 5 : Validation Accuracy = 89.728%	EPOCH 5 : Validation Accuracy = 89.592%
EPOCH 6 : Validation Accuracy = 90.748%	EPOCH 6 : Validation Accuracy = 94.150%
EPOCH 7 : Validation Accuracy = 91.179%	EPOCH 7 : Validation Accuracy = 96.168%
EPOCH 8 : Validation Accuracy = 92.358%	EPOCH 8 : Validation Accuracy = 97.143%
EPOCH 9 : Validation Accuracy = 93.515%	EPOCH 9 : Validation Accuracy = 98.050%
EPOCH 10: Validation Accuracy = 92.902%	**EPOCH 10: Validation Accuracy = 97.664%**
EPOCH 11: Validation Accuracy = 92.585%	EPOCH 11: Validation Accuracy = 98.458%
EPOCH 12: Validation Accuracy = 92.721%	EPOCH 12: Validation Accuracy = 97.914%
EPOCH 13: Validation Accuracy = 91.633%	EPOCH 13: Validation Accuracy = 98.231%
EPOCH 14: Validation Accuracy = 92.041%	EPOCH 14: Validation Accuracy = 98.390%
EPOCH 15: Validation Accuracy = 93.288%	EPOCH 15: Validation Accuracy = 98.889%
EPOCH 16: Validation Accuracy = 93.515%	EPOCH 16: Validation Accuracy = 98.141%
EPOCH 17: Validation Accuracy = 93.537%	EPOCH 17: Validation Accuracy = 98.776%
EPOCH 18: Validation Accuracy = 91.859%	EPOCH 18: Validation Accuracy = 98.503%
EPOCH 19: Validation Accuracy = 93.719%	EPOCH 19: Validation Accuracy = 98.594%
EPOCH 20: Validation Accuracy = 93.401%	EPOCH 20: Validation Accuracy = 98.753%
EPOCH 21: Validation Accuracy = 94.354%	EPOCH 21: Validation Accuracy = 98.594%
EPOCH 22: Validation Accuracy = 93.810%	EPOCH 22: Validation Accuracy = 99.138%
EPOCH 23: Validation Accuracy = 93.923%	EPOCH 23: Validation Accuracy = 98.753%
EPOCH 24: Validation Accuracy = 93.356%	EPOCH 24: Validation Accuracy = 99.184%
EPOCH 25: Validation Accuracy = 93.673%	EPOCH 25: Validation Accuracy = 99.025%
EPOCH 26: Validation Accuracy = 93.515%	EPOCH 26: Validation Accuracy = 99.297%
EPOCH 27: Validation Accuracy = 94.558%	EPOCH 27: Validation Accuracy = 98.571%
EPOCH 28: Validation Accuracy = 93.832%	EPOCH 28: Validation Accuracy = 99.070%
EPOCH 29: Validation Accuracy = 93.832%	EPOCH 29: Validation Accuracy = 98.889%
EPOCH 30: Validation Accuracy = 93.447%	EPOCH 30: Validation Accuracy = 99.184%

Tabla 4.3 Configuración y resultados de las redes VGGNet y LeNet

Como se observa en la tabla 4.3 durante los diferentes recorridos, ambos modelos se han saturado después de solo 10 épocas, por lo que se podría reducir los epoch a 10 y ahorrar recursos computacionales. La precisión final en cada modelo ha sido:

- LeNet ha alanzado una precisión de validación máxima del 93,4%.
- VGGNet ha alcanzado una precisión de validación máxima del 99,1%.

Para la tarea de clasificación de señales, **se ha elegido la red VGGNet** al tener 5,7% más de precisión en el conjunto de validación que la red LeNet.

4.8 Pruebas del modelo

4.8.1 Uso en ejemplos de prueba

Para calcular la precisión del modelo VGGNet sobre ejemplos desconocidos se usarán los ejemplos de prueba, los cuales no han sido usados previamente, estos se componen de 4410 imágenes.

El resultado obtenido es:

- **Test Accuracy = 97,6%**

Es un resultado correcto, aunque no es perfecto. Para ver los casos donde el modelo no es capaz de predecir la etiqueta correctamente se traza la matriz de confusión, que es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que se puede comprobar si el sistema está confundiendo dos clases.

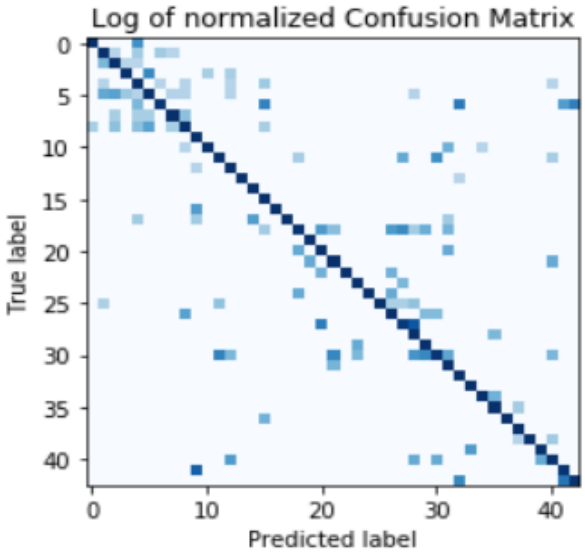


Figura 4.18 Matriz de confusión

En la matriz de confusión se pueden observar dos grupos de predicciones que han sido erróneamente calificadas.

- El primero de ellos se encuentra en la zona superior-izquierda de la figura 4.18 a causa de los diversos límites de velocidad, que a veces se clasifican erróneamente entre ellos, debido a que al tratarse de dígitos es fácil que la red los confunda entre ellos.
- El segundo grupo se puede observar es en la zona central de la figura 4.18, debido a las señales de tráfico con forma triangular pueden clasificarse erróneamente entre ellas a causa de los dibujos que se encuentran en el interior, siempre no son fácilmente identificables, haciendo que la red confunda unas con otras.

4.8.2 Uso en nuevas imágenes

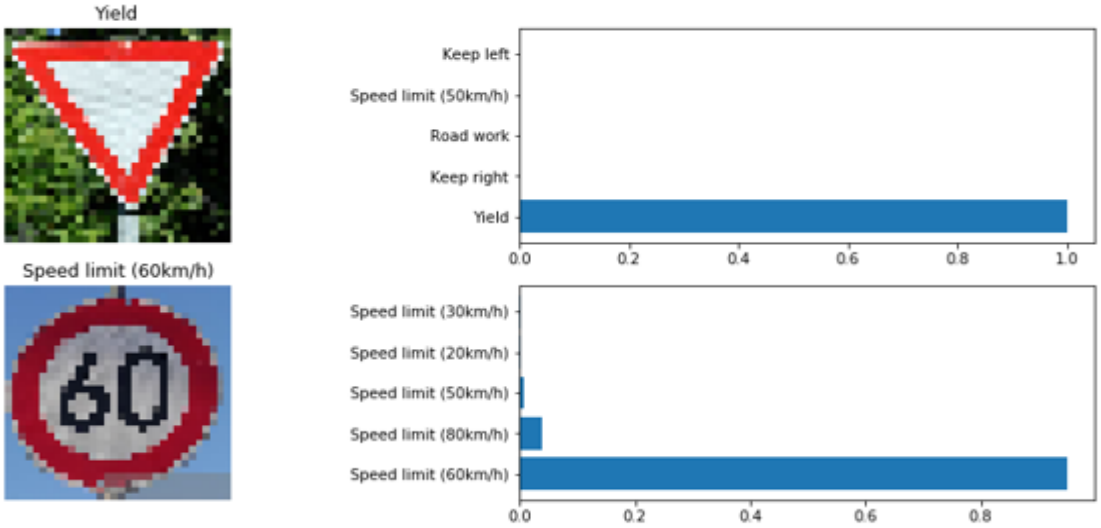
En este paso, se usará el modelo para predecir la clase de 10 de señales de tráfico aleatorias sacadas de la web para comprobar el rendimiento.

Las imágenes usadas se pueden ver en la figura 4.19.

Figura 4.19 Imágenes nuevas usadas para comprobar el rendimiento de la CNN

Estas imágenes de prueba incluyen algunas señales que pueden ser fáciles de predecir, y otras que se consideran difíciles de predecir para el modelo. Por ejemplo, las señales fáciles de predecir como "Stop" y "No entry", las cuales son claras y además pertenecen a las clases donde el modelo puede predecir con alta precisión. Por otro lado, tenemos señales que pertenecen a clases donde hay poca precisión, como la señal de "Speed Limit", porque como se pudo ver en la matriz de confusión, resulta que los diversos límites de velocidad a veces se clasifican erróneamente entre ellos, lo mismo sucede con la señal de "Pedestrians", debido a que las señales de tráfico con forma triangular pueden ser confundidas entre sí.

En la figura 4.20 y 4.21 se muestran los resultados obtenidos, en ellas se puede observar las 5 probabilidades más altas que ha obtenido el modelo y la confianza en cada predicción.



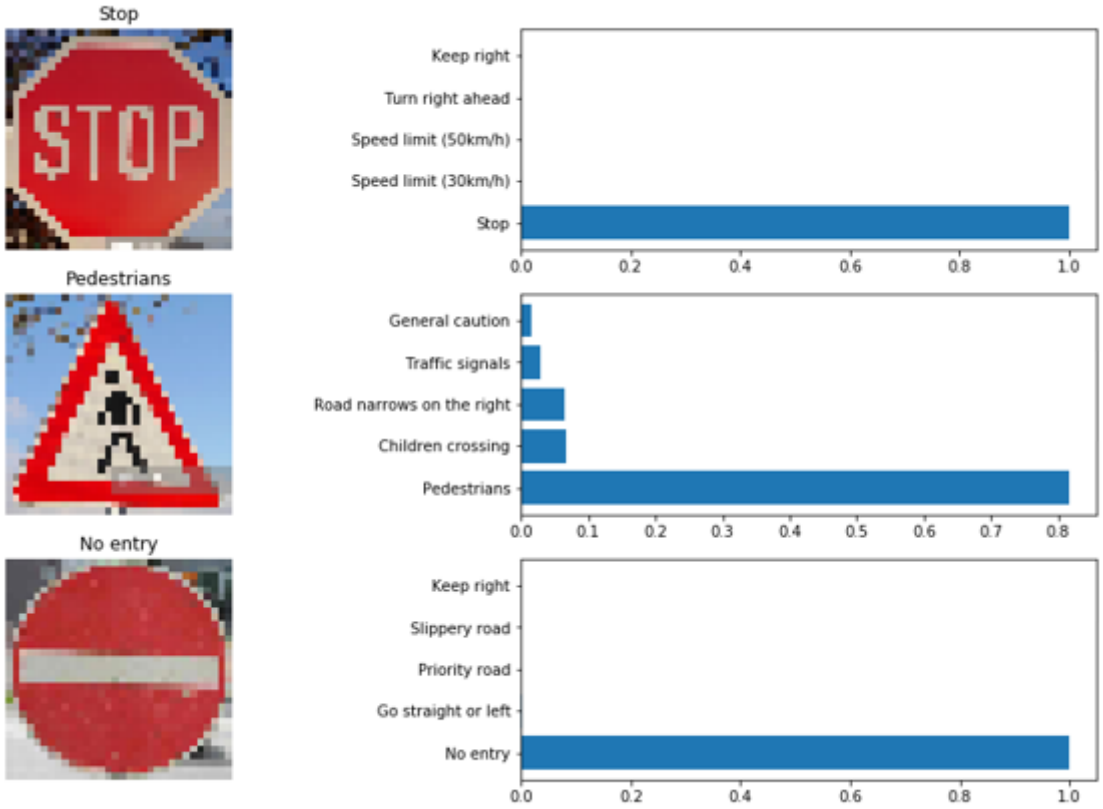


Figura 4.20 Resultados del reconocimiento de nuevas imágenes por la CNN

Como se puede observar en la figura 4.20, en las probabilidades más altas que se han obtenido, el modelo alcanza una confianza muy alta (100%) cuando se trata de predecir signos simples, como el signo "Stop" o "No entry", e incluso es capaz de predecir con un 100% un signo triangular simple, como el signo "Yield". Por otro lado, la confianza del modelo se reduce ligeramente con el signo triangular más complejo como la señal “Pedestrians", además de ser una imagen bastante ruidosa, se trata de un signo triangular con una forma en su interior, el modelo fue capaz de predecir la clase, pero con un 80% de confianza. Y en el signo "Speed Limit", se puede observar que el modelo predice con precisión que se trata de un signo de límite de velocidad, pero de alguna manera se confunde entre los diferentes límites de velocidad.

En la figura 4.21, se observa que el modelo es capaz de clasificar correctamente las señales con forma triangular como son las señales de “beware of ice/snow” y “General Caution”, estas tienen un dibujo claro en su interior de manera que el modelo es capaz de identificarlos con más del 95% de confianza. Sin embargo, en las señales de velocidad es donde el modelo tiene más problemas, de los tres casos expuestos en la figura 4.21 solo ha sido capaz de identificar el signo “Speed limit” de 120 km/h con un 100% de confianza, mientras que en los signos de velocidad de 50 y 70 km/h no ha sido capaz de identificar correctamente el límite de velocidad, pero en ambos casos sí que ha identificado correctamente que se trata de señales de velocidad.

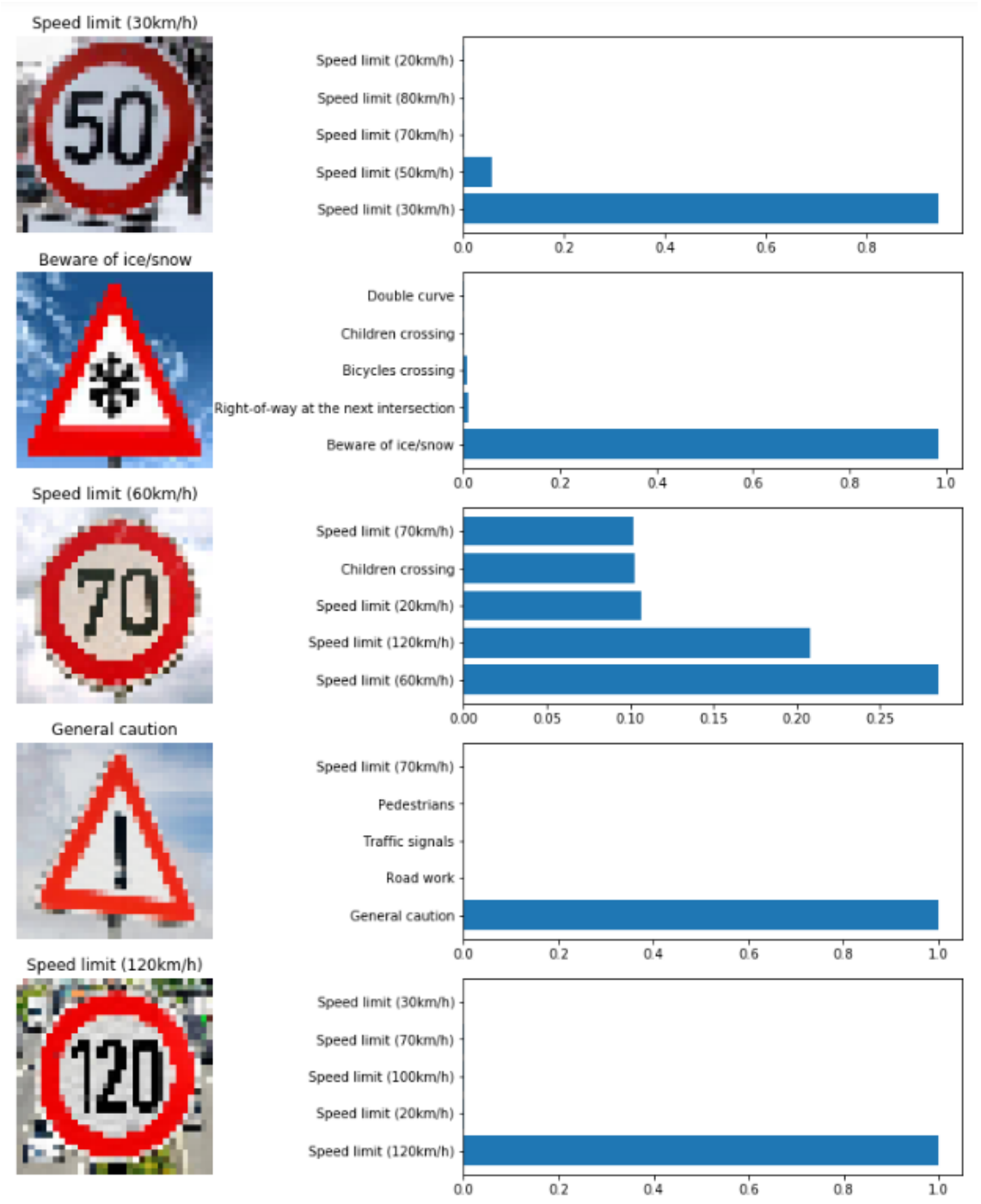


Figura 4.21 Resultados del reconocimiento de nuevas imágenes por la CNN

El modelo VGGNet pudo predecir la clase correcta, aunque no exacta como en los casos de las señales de velocidad, para cada una de las 10 nuevas imágenes de prueba. Con una exactitud de prueba del 80%, se puede decir que el modelo es seguro.

Con el modelo VGGNet, se ha podido alcanzar una tasa de precisión alta. Se puede observar que los modelos se saturan después de casi 10 épocas, por lo que se podrían ahorrar algunos recursos computacionales y reducir el número de épocas a 10. Además, se puede probar otras técnicas de preprocesamiento para mejorar aún más la precisión del modelo.

Se podría mejorar aún más el modelo usando CNN jerárquicos para identificar primero grupos más amplios (como señales de velocidad) y luego tener CNN para clasificar características más finas (como el límite de velocidad real).

Este modelo solo funcionará en ejemplos de entrada donde las señales de tráfico se centran en el centro de la imagen. No tiene la capacidad necesaria para detectar señales que se encuentren en las esquinas de la imagen.

CAPÍTULO 5

IMPLEMENTACIÓN DEL SISTEMA DE DETECCIÓN DE SEÑALES EN UN SISTEMA AUTÓNOMO

5.1 Introducción

En este capítulo se presentará el proceso llevado a cabo para implementar el sistema de detección de señales de tráfico en un sistema autónomo capaz de instalarse en un vehículo autónomo.

5.2 Elección de cámara

A la hora de usar una cámara para el proyecto es importante que sea compatible con la plataforma en la que se ha desarrollado, por ello se ha elegido la cámara de Intel RealSense D415, mostrada en la figura 5.1, la cual soporta una amplia variedad de sistemas operativos y lenguajes de programación.

Figura 5.1 Intel RealSense D435

El Intel RealSense SDK 2.0 permite extraer los datos de la cámara e interpretarlos en diferentes plataformas, ya sea Windows, Linux, macOS, etc. El SDK proporciona código de ejemplo de software libre para varios lenguajes como Python, Node.js, C#/.NET y C/C++. También tiene la posibilidad de integrar con tecnologías de terceros como ROS, Unity, OpenCV, PCL y Matlab, así como software adicional de Intel RealSense.

Esta cámara es capaz de obtener la profundidad de la escena, imágenes de calidad en RGB y elevada frecuencia de fotogramas, como se observa en la figura 5.2 y 5.3.

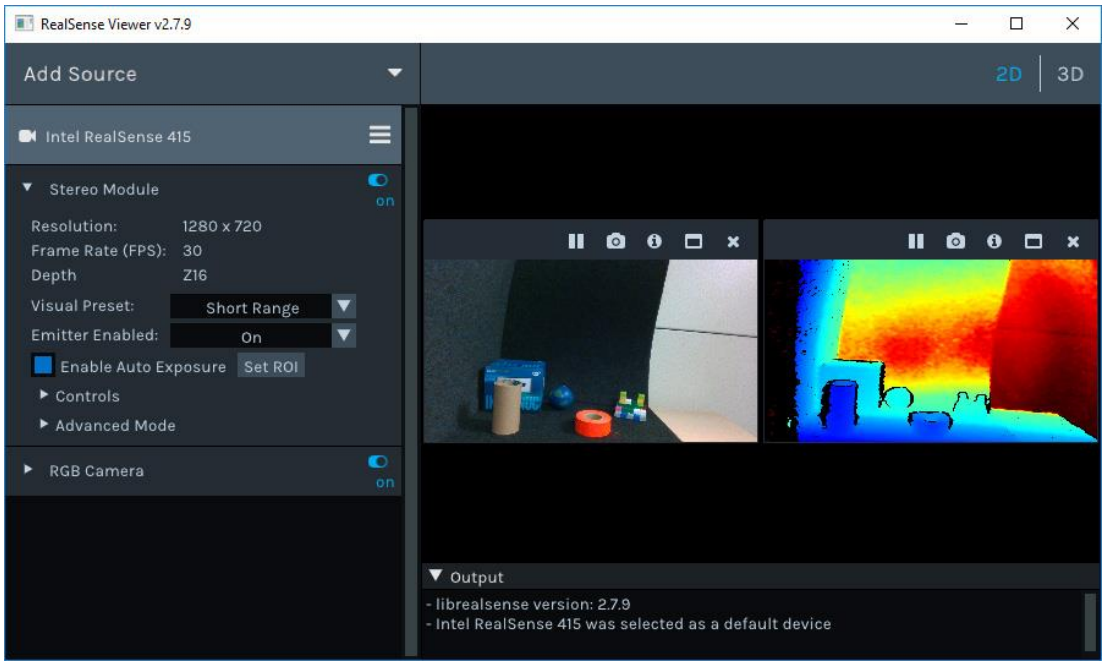


Figura 5.2 Imagen RGB y de profundidad obtenida por la cámara RealSense

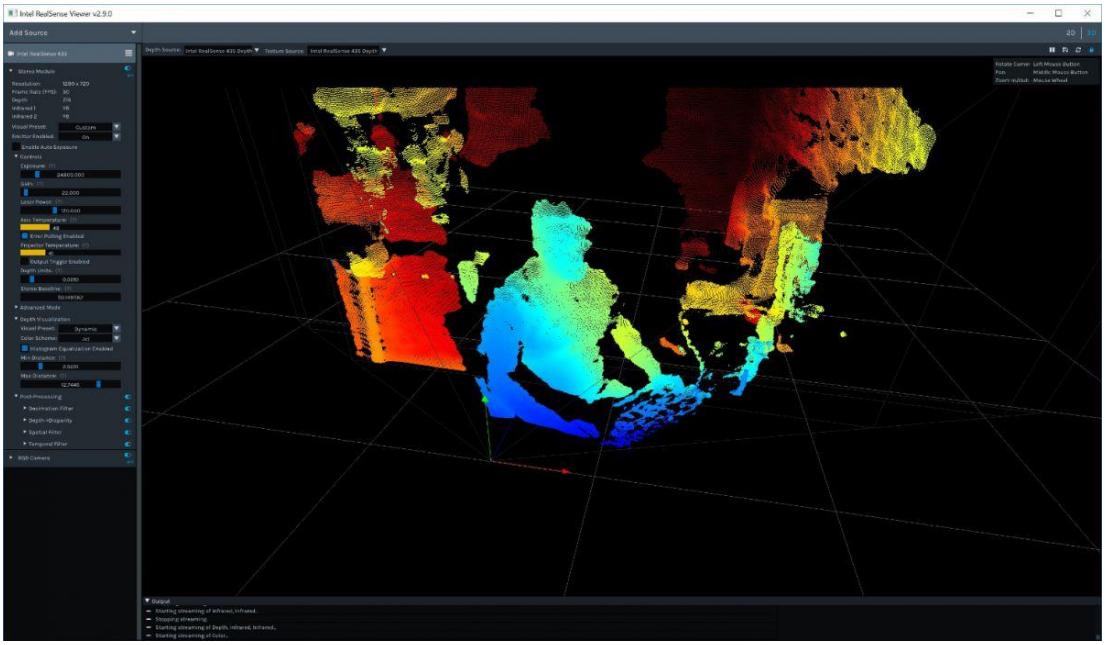


Figura 5.3 Nube de puntos obtenido mediante la cámara RealSense

Las principales características [35] de la cámara son:

- Grabación de imágenes RGB con una resolución máxima de 1920x1080 pixeles y hasta 30 imágenes por segundo.
- Sensor RGB FOV(HxVxD): 69.4° x 42.5° x 77° (+/- 3°).
- Grabación de imágenes de profundidad con una resolución máxima de 1280x720 pixeles y hasta 90 imágenes por segundo.
- Tecnología de captura de profundidad: Active IR Stereo con una distancia mínima de 0.105m y una máxima de 10 m (Depende del entorno).
- Longitud x profundidad x altura: 90 mm x 25 mm x 25 mm.
- Conexión mediante USB C 3.1.

5.3 Algoritmos de imágenes piramidales y ventana deslizante

Una de las formas de reconocer objetos en una imagen es mediante el uso de los algoritmos de imágenes piramidales y ventana deslizantes. A partir de ellos se pueden crear clasificadores que puedan reconocer objetos a diferentes escalas y ubicaciones en las imágenes.

La imagen piramidal es una representación a multiescala de la imagen, esto permite encontrar objetos en imágenes a diferentes escalas y al ser combinado con una ventana deslizante se podría encontrar objetos en imágenes en varios lugares.

En la figura 5.4 se muestra como sería una imagen piramidal, en la parte inferior de la pirámide se encuentra la imagen original que tiene su tamaño original (en términos de ancho y alto). Y en cada capa posterior, la imagen se redimensiona. La imagen es redimensionada progresivamente hasta que se cumpla algún criterio de detención, que normalmente es un tamaño mínimo y no es necesario realizar más submuestreo.

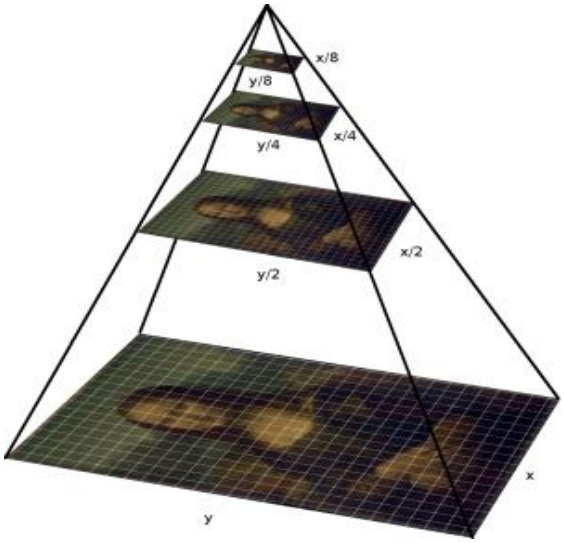


Figura 5.4 Ejemplo de una imagen piramidal

En el contexto de la visión por computadora una ventana deslizante es una región rectangular de ancho y altura fijos que "se desliza" a través de una imagen. Para cada una de estas ventanas, se tomará la región de la ventana y se aplica un clasificador de imágenes para determinar si la ventana tiene el objeto que se busca, en el caso del proyecto se buscará en ella la señal de tráfico.

Ambas técnicas son usadas para la detección de objetos y la clasificación de imágenes. En proyecto se han implementado ambas técnicas para la búsqueda de señales en una imagen.

5.4 Implementación de un sistema de detección de señales

La detección de señales de tráfico es una herramienta muy útil para sistemas de conducción autónoma. Para la implementación del sistema de detección de señales se usará el modelo VGGNet entrenado durante la consecución del proyecto.

El proceso se ha dividido en una serie de pasos:

1) Captura de imagen. La información del entorno se obtiene con la cámara D435, que proporciona 6 imágenes por segundo en espacio de color RGB de 640x480 pixeles. La combinación de resolución y de imágenes por segundo dependerá del hardware donde se ejecute el programa, debido a que los cálculos que se realizan en el algoritmo son muy complejos, será necesario un PC con altas prestaciones para que no se produzca un retraso considerable entre la imagen obtenida por la cámara y la imagen mostrada por pantalla.
2) Ventaneo y escalado. En esta etapa, se aplica el algoritmo de ventana deslizante y de escalado piramidal sobre la imagen capturada por la cámara. El tamaño de ventana y el número de veces que se redimensiona la imagen dependerá del número de ventanas que se quieran obtener, es decir, si se usan ventanas grandes como de 400x400 pixeles con un paso de 50 pixeles entre ellas se extraen únicamente 10 ventanas, mientras que, si se una un tamaño de 40x40 pixeles se obtienen más de 100 ventanas que analizar, esto hace que el proceso sea más lento lo que provoca un retraso considerable entre la imagen obtenida y la imagen mostrada por pantalla.

3) Preprocesado. En tercer lugar, las ventanas obtenidas tienen que ser redimensionadas al tamaño de 32x32 pixeles, que es el tamaño de las imágenes con el que ha sido entrenada la red, además la ventana tendrá que seguir el preprocesado que recibieron las imágenes que fueron utilizadas para entrenar la red VGGNet. Para ello la ventana será convertida a escala de grises, se ecualizará su histograma local y por último se normalizará. Una vez realizados estos procesos la ventana estará lista para ser entregada a la red.
4) Clasificación. En esta etapa la ventana preprocesada es entregada a la red neuronal y esta devuelve dos datos, Y_PROB que es el porcentaje de confianza (1 máxima confianza) y el segundo, Y_PRED que es la clase de señal a la que pertenece entre las 43 posibles clases.
5) Representación gráfica. Si Y_PROB es mayor o igual del 90% entonces la imagen tomada por la cámara se mostrará en pantalla con un recuadrado que marcará la zona donde la señal es encontrada. Además, en el recuadro se integra la clase de señal en la que ha sido clasificada.
6) El proceso se repetirá volviendo al paso uno donde se capturará una imagen con la cámara y se realizarán de nuevo todos los pasos mientras que el programa no sea detenido.

En el flujograma de la figura 5.5, se puede ver de forma esquemática la implementación del sistema de detección de tráfico.

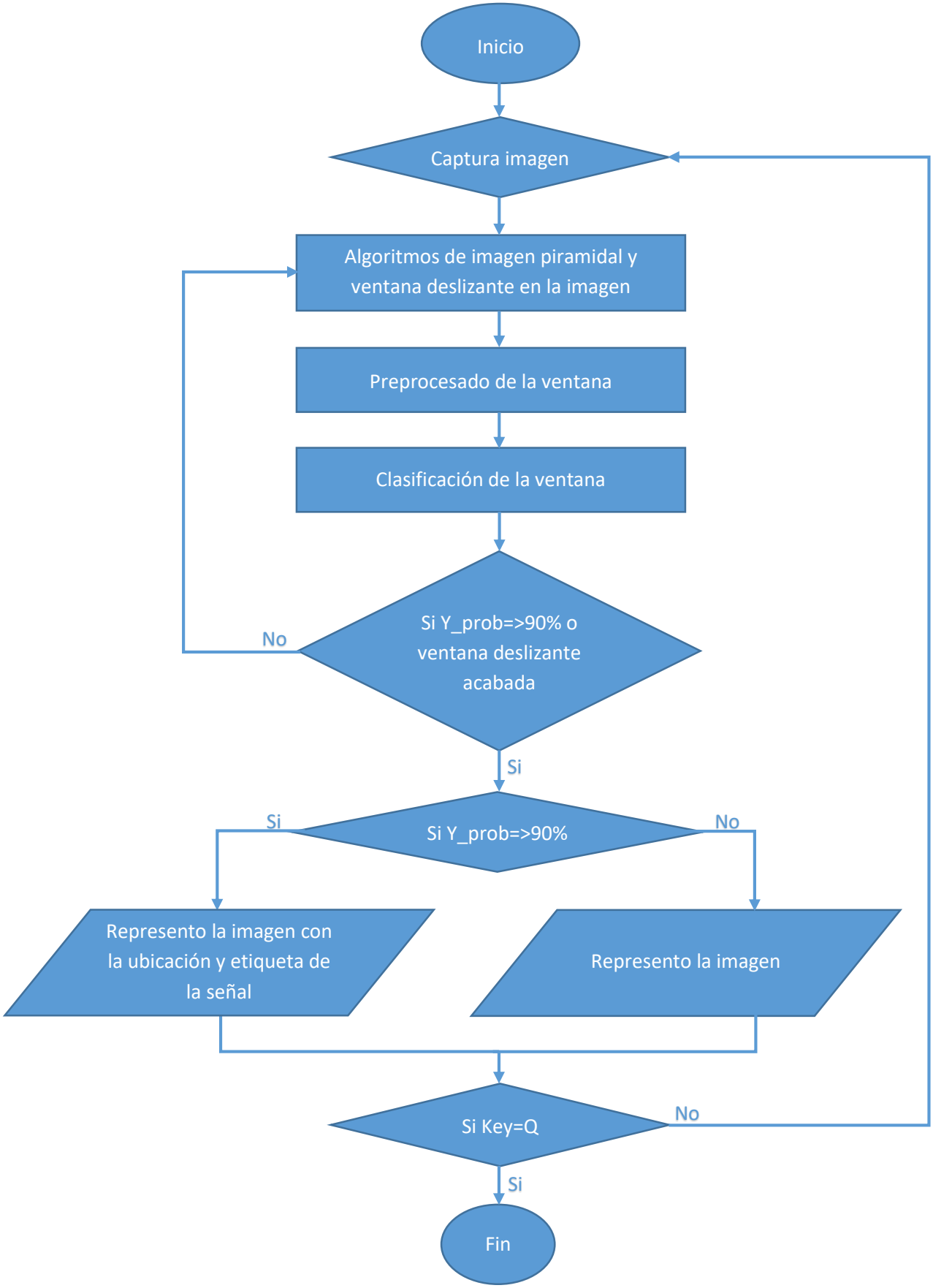


Figura 5.5 Flujograma del sistema de detección de señales

5.5 Pruebas del algoritmo

Para comprobar el funcionamiento se ha instalado todo el software necesario en un PC con las siguientes especificaciones:

- CPU: Intel Core i7-4790K 4.0Ghz Box Socket 1150
- GPU: Gigabyte GeForce GTX 970 WindForce 3X OC 4GB DDR5
- RAM: G.Skill Ripjaws X DDR3 1866 PC3-14900 16GB 2x8GB CL9

Se han impreso en 3D modelos de señales de tráfico para hacer pruebas de la precisión del algoritmo en el laboratorio, las cuales han sido obtenidas en la web [36],en la figura 5.6 se muestran.

Se trata de modelos que han sido diseñados para ser proporcionalmente precisos.

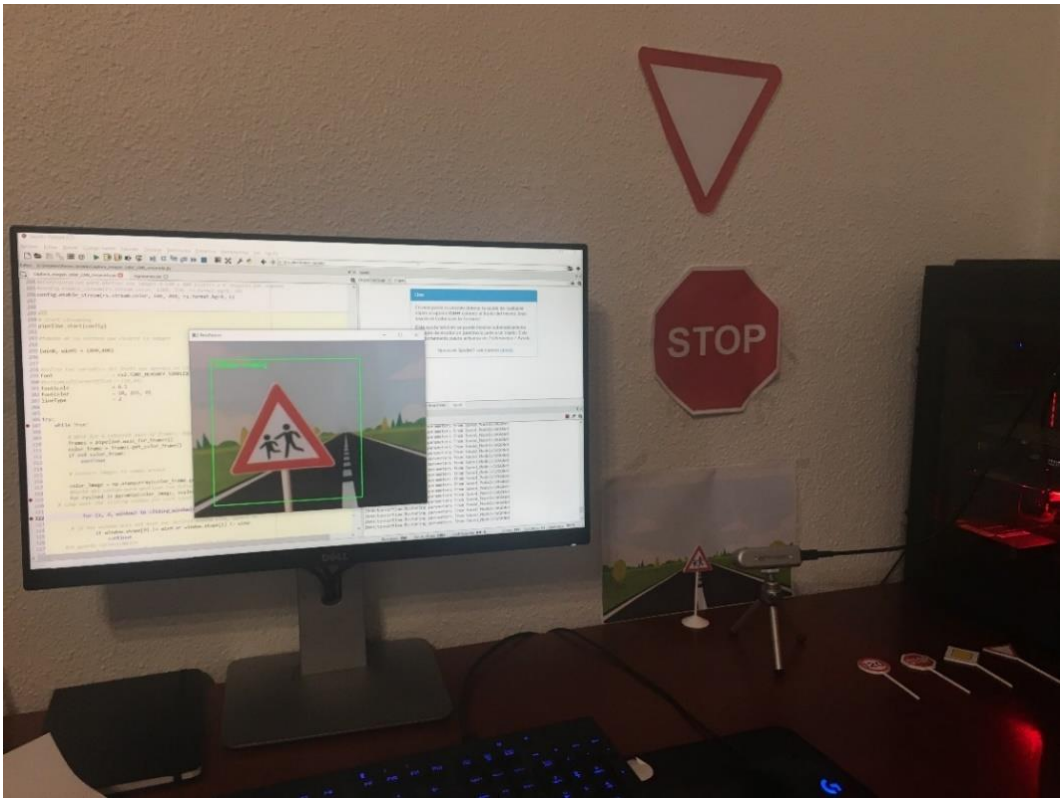


Figura 5.6 Pruebas del sistema de detección de señales en laboratorio

La prueba realizada consiste en que el programa diseñado, sin ninguna intervención sea capaz de detectar y etiquetar 6 señales con formas circular, triangular, romboide y hexagonal. Para simular condiciones reales se ha usado un fondo de carretera.

Los resultados obtenidos con las diferentes señales son mostrados en las siguientes figuras 5.7, 5.8, 5.9, 5.10, 5.11 y 5.12.

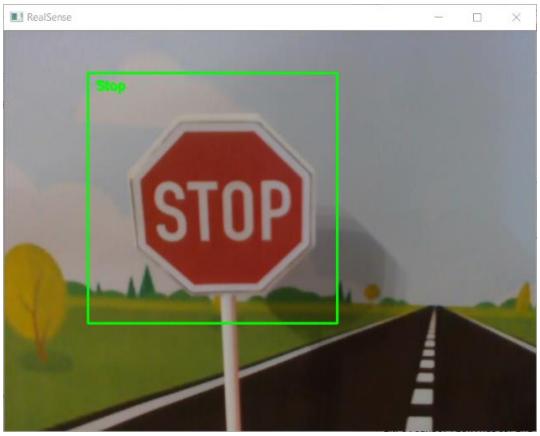


Figura 5.7 Reconocimiento señal de stop

Figura 5.8 Reconocimiento señal de velocidad

Figura 5.9 Reconocimiento señal de prioridad

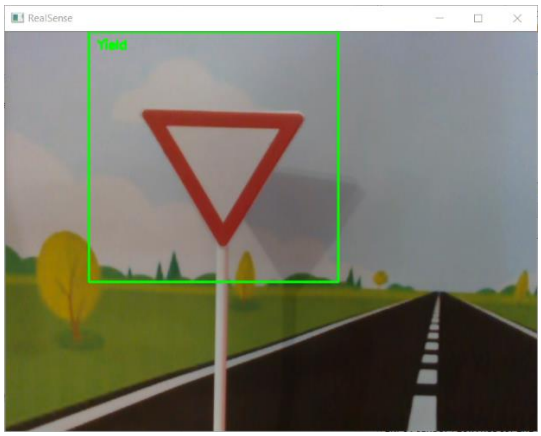


Figura 5.10 Reconocimiento señal de ceda el paso

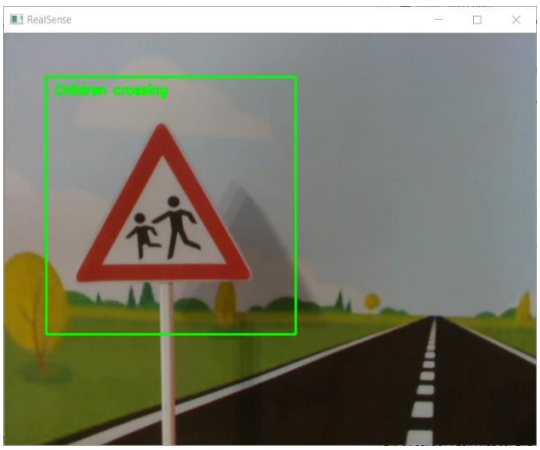


Figura 5.11 Reconocimiento señal de niños cruzando

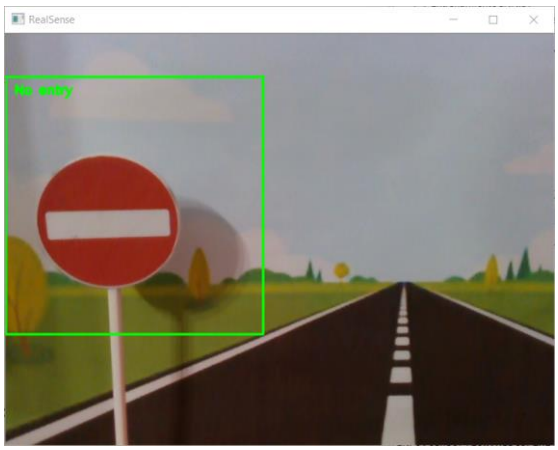


Figura 5.12 Reconocimiento señal prohibido el paso

En las seis pruebas realizadas el porcentaje de acierto obtenido ha sido del 100%, aunque a veces se puede obtener un falso positivo al mover las señales para posicionarlas delante de la cámara una vez que se coloca en su posición el algoritmo ha sido capaz de etiquetarlas correctamente.

A la hora de ejecutar el programa, se obtiene un retraso entre la imagen capturada y la imagen de salida de entre 2 a 5 segundos, siempre que se use un tamaño de ventana grande,

que en la prueba ha sido de 300x300, ya que si se extrajeran más ventanas para su clasificación el tiempo de retraso entre la imagen capturada y la imagen mostrada por pantalla se dispara.

La inteligencia artificial está teniendo un gran impacto en nuestra sociedad y se pueden encontrar hardware mucho más potentes que están específicamente diseñados para usar redes neuronales, haciendo que el tiempo entre la imagen tomada y la mostrada por pantalla sea casi inapreciable, además se podrían usar ventanas mucho más pequeñas, usar imágenes con una resolución superior y analizar más imágenes por segundo.

CAPÍTULO 6

CONCLUSIÓN Y TRABAJOS FUTUROS

6.1 Introducción

En este capítulo se presentarán las conclusiones obtenidas durante la ejecución del proyecto, así como los trabajos futuros derivados del mismo.

6.2 Conclusión

Las conclusiones que se han obtenido en este trabajo son:

- ✓ Se ha realizado un estado del arte sobre la tecnología de visión por computador, se han presentado las principales funciones necesarias para la detección de objetos para una conducción autónoma, como pueden ser las líneas viales, además se ha realizado una introducción sobre el aprendizaje profundo que, a pesar de ser un tema bastante complejo, se han dado los conocimientos necesarios para entender el presente trabajo.

- ✓ Se ha llevado a cabo un desarrollo en lenguaje de programación Python, el más utilizado en la actualidad a nivel ingenieril, usando en el entorno Spyder. Además, se han integrado en el proyecto librerías de funciones como OpenCV para la visión artificial y Tensorflow para el desarrollo de las redes neuronales, además de otras.

- ✓ En este trabajo se han desarrollado dos métodos de detección de líneas viales. El primero de ellos, se basa en el cambio del espacio de color de manera que en la imagen solo se conserven las zonas de interés, a las cuales se someterá a funciones de trazado de contornos y líneas, lo que hace posible detectar la vía. Este algoritmo ofrece su mejor rendimiento sobre escenas con buenas condiciones lumínicas y con sus contornos claramente marcados. El segundo algoritmo desarrollado se basa en el cambio de perspectiva mediante la transformación de la imagen a vista cenital. La vista cenital permite que las líneas curvas sean más fáciles de detectar. La detección de las líneas se lleva a cabo mediante el cálculo del histograma y a través de una ventana deslizante.

- ✓ Se ha desarrollado un sistema de detección de señales de tráfico basado en CNN capaz de detectar 43 clases distintas de señales. Para determinar el sistema con mejor rendimiento en la detección se ha realizado un estado del arte de las redes neuronales convolucionales, se han expuesto las CNN más utilizadas en la actualidad y de dicho estudio se han implementado las redes LeNet y VGGNet para el propósito del trabajo.

- ✓ Para determinar la red con mayor rendimiento (VGGNet o LeNet), sus arquitecturas han sido entrenadas con un dataset de señales de tráfico alemanas que contiene 50000 imágenes de 43 clases diferentes, las cuales han sido redimensionadas a 32x32 pixeles. Después de entrenar las redes con el dataset se ha obteniendo una precisión de validación del 93,4% en la red LeNet y 99,1% en la red VGGNet. Esta última ha sido seleccionada para el proyecto al obtener un rendimiento mayor (5,7% más que la red LeNet).

- ✓ Para comprobar la precisión del modelo VGGNet en imágenes que no hayan sido usadas para el entrenamiento ni para la validación, se ha testado con el conjunto de prueba, que contiene 4410 imágenes, obteniendo un porcentaje de acierto del 97,6%. Además, se ha obtenido la matriz de confusión en la que se ha observado cuales son las señales donde más problemas de clasificación tiene el modelo. La clase de señales de límite de velocidad a veces puede ser clasificada erróneamente esto se debe a que los dígitos en su interior pueden ser confundidos entre sí, lo mismo sucede en las señales con forma triangular que pueden clasificarse erróneamente entre ellas, al confundirse unas con otras.

- ✓ Un problema que se ha encontrado cuando se ha tratado de clasificar señales de velocidad españolas, es que son ligeramente diferentes a las alemanas, las cuales tienen un contorno blanco que no tienen las españolas lo que confunde al modelo.

- ✓ Por último, se ha desarrollado una aplicación en Python que permite cargar el modelo VGGNet entrenado y es capaz a través de una cámara de obtener una imagen del entorno. La aplicación permite detectar en las imágenes obtenidas las 43 señales para las que se entrenó la red VGGNet. Para ello, en cada imagen capturada por la cámara se aplica un algoritmo de ventana deslizante y de escalado piramidal en busca de porciones de imagen que contengan una señal de tráfico. El programa ha sido testado con seis señales de tráfico a escala extraídas de la web y ha sido capaz de identificar correctamente el 100% de ellas. El programa necesita ser optimizado para su ejecución en un GPU ya que emplea un tiempo excesivo en la búsqueda las señales en la imagen.

Para acabar se puede concluir que se han desarrollado un sistema de detección de líneas viales, un sistema de detección de señales de tráfico y se ha implementado un sistema autónomo de detección de señales de tráfico. Así que se puede decir que se han cumplido todos los objetivos propuestos para el trabajo.

6.3 Trabajos futuros

A continuación, se presentan una serie de líneas o proyectos que podrían realizarse a partir del trabajo realizado:

- Incrementar el número de muestras de la CNN. No existe una cantidad óptima, pero posiblemente aumentará la precisión y la generalidad cuando más ejemplos disponga en particular en los casos de las señales que más son confundidas, que como se vio en la matriz de confusión son los casos de las señales de velocidad y las señales con forma triangular.
- Se podría mejorar aún más la CNN de detección de señales de tráfico usando CNN jerárquicos para identificar primero grupos más amplios (como señales de velocidad) y luego tener CNN para clasificar características más exactas (como el límite de velocidad real).

- Diseñar una red neuronal para detectar líneas viales siendo capaz de funcionar en condiciones desfavorables, como por ejemplo, cuando sus contornos no se encuentren bien definidos.
- Realizar una implementación del detector de líneas y del detector de señales en un vehículo como un asistente de ayuda a la conducción (ADAS), como ya usan muchos vehículos o implementarlo en sistema de conducción autónoma.

Bibliografía

[1] A. L. Kun, S. Boll, and A. Schmidt, “Shifting Gears: User Interfaces in the Age of Autonomous Driving,” *IEEE Pervasive Comput.*, 2016.

[2] T. Huang, “Computer Vision: Evolution And Promise,” *19th Cern Sch. Comput.*, pp. 21–25, 1996.

[3] R. Gonzalez and R. Woods, *Digital image processing*. 2002.

[4] A. Gonzalez, *Técnicas y algoritmos basicos de vision artificial*. 1989.

[5] S. Sural, G. Qian, and S. Pramanik, “Segmentation and histogram generation using the HSV color space for image retrieval,” in *IEEE International Conference on Image Processing*, 2002.

[6] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 1986.

[7] J. Valverde-Rebaza, “Detección de bordes mediante el algoritmo de Canny.,” 2007.

[8] “Operador Sobel - Wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/Operador_Sobel. [Accessed: 07-Nov-2019].

[9] L. López-Martínez, José & Canul, Luis & Narváez-Díaz, “Un algoritmo rápido de la transformada de Hough para la detección de líneas rectas en una imagen,” 2015.

[10] A. L. Samuel, “Some Studies in Machine Learning,” *IBM J. Res. Dev.*, 1959.

[11] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*. 2015.

[12] J. Luo and J. Huang, “Generative adversarial network: An overview,” *Yi Qi Yi Biao Xue Bao/Chinese Journal of Scientific Instrument*. 2019.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[14] “Redes Neuronales, ¿qué son?. - William Khepri - Medium.” [Online]. Available: https://medium.com/@williamkhepri/redes-neuronales-que-son-a64d022298e0. [Accessed: 06-Nov-2019].

[15] “Red neuronal artificial - Wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/Red_neuronal_artificial. [Accessed: 06-Nov-2019].

[16] C. Thorpe, M. Hebert, T. Kanade, and S. Shaffer, “Toward Autonomous Driving: The CMU Navlab Part I — Perception,” *IEEE Expert. Syst. their Appl.*, 1991.

[17] E. D. Dickmanns and B. D. Mysliwetz, “Recursive 3-D Road and Relative Ego-State Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 1992.

[18] “OpenCV.” [Online]. Available: https://opencv.org/. [Accessed: 25-Sep-2019].

[19] “Anaconda | The World’s Most Popular Data Science Platform.” [Online]. Available: https://www.anaconda.com/. [Accessed: 25-Sep-2019].

[20] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, “Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks,” pp. 1–12, 2019.

[21] “German Traffic Sign Benchmarks.” [Online]. Available: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset. [Accessed: 09-Oct-2019].

[22] P. Sermanet and Y. Lecun, “Traffic sign recognition with multi-scale convolutional networks,” in *Proceedings of the International Joint Conference on Neural Networks*, 2011.

[23] “Convolutional Neural Networks for Visual Recognition.” [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 16-Oct-2019].

[24] G. Wei, G. Li, J. Zhao, and A. He, “Development of a LeNet-5 gas identification CNN structure for electronic noses,” *Sensors (Switzerland)*, vol. 19, no. 1, pp. 1–17, 2019.

[25] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, 1998.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012.

[28] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[29] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[30] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” pp. 1–14, 2014.

[31] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[32] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning, ICML 2015*, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[34] “TensorFlow.” [Online]. Available: https://www.tensorflow.org/. [Accessed: 09-Dec-2019].

[35] “Depth Camera D435 – Intel® RealSense™ Depth and Tracking Cameras.” [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/. [Accessed: 26-Nov-2019].

[36] “Descargar STL gratis Modelos de señales de tráfico · Cults.” [Online]. Available: https://cults3d.com/es/modelo-3d/juegos/traffic-sign-models. [Accessed: 28-Nov-2019].