UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – PROCESAMIENTO DIGITAL DE SEÑALES

MONITOREO DE CULTIVOS A TRAVÉS DEL ANÁLISIS DE IMÁGENES MEDIANTE IA PARA DETECCIÓN DE CULTIVOS DE RIEGOS

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
LUIS ANDRÉS ROBLERO GONZÁLEZ

TUTOR (ES) PRINCIPAL(ES)
DR. MIGUEL MOCTEZUMA FLORES /
Facultad de Ingeniería

CIUDAD UNIVERSITARIA, CD. FEBRERO 2023

JURADO ASIGNADO:

Presidente:	Dr. Miguel Moctezuma Flores
Secretario:	Dra. Jimena Olveres Montiel
1 er. Vocal:	Dr. Jesús Savage Carmona
2 do. Vocal:	Dr. Miguel A. Padilla Castañeda
3 er. Vocal:	Dr. Pablo Roberto Pérez Alcázar.

Lugar o lugares donde se realizó la tesis: CIUDAD UNIVERSITARIA, CD. MX

TUTOR DE TESIS:

DR. MIGUEL MOCTEZUMA FLORES

FIRMA

El desarrollo de esta tesis "*Monitoreo De Cultivos A Través Del Análisis De Imágenes Mediante IA Para Detección De Cultivos De Riegos"*, fue desarrollada en un 50% de manera virtual con Doctores y maestros de la Universidad Nacional Autónoma de México durante el periodo 2020-2022 del programa de Posgrado; ya que, debido al brote del virus Covid-19, no se pudo trabajar en los laboratorios de la Universidad directamente, el otro 50% del trabajo fue realizado en el campo, directamente en plantíos de cultivo de maíz para poder observar y analizar de manera más cercana el proceso de cultivación y comportamiento de la planta del maíz en un ambiente controlado en los terrenos de cultivo del Ejido Verapaz, ubicado en el municipio de Frontera Comalapa del estado de Chiapas, todo esto, bajo la tutoría de: DR. Miguel Moctezuma Flores

AGRADECIMIENTOS A:

Universidad Nacional Autónoma de México

Instituto de Ciencias Aplicadas y Tecnología
La Facultad de Ingeniería

Programa de Becas CONACYT

Sistema de riego del C. Francisco Juárez

Sistema de riego del C. Dagoberto Roblero

A los campos de cultivo del ejido Verapaz del estado de Chiapas

Director del ICAT -UNAM Dr. Rodolfo Zanella Specia

Coordinador del Programa de Maestría y Doctorado en Ingeniería Dr. Alfonso Durán Moreno

Mis tutores:
DR. MIGUEL MOCTEZUMA FLORES

Comité de Evaluación y Asesorías:
Dr. Jesús Savage Carmona,
Dra. Jimena Olveres Montiel,
Dr. Caleb Antonio Rascón Estebané,
Dr. Miguel A. Padilla Castañeda,
Dr. Pablo Roberto Pérez Alcázar,
Dra. Graciela Velasco,
Dra. Lucía Medina Gómez.

Este trabajo fue apoyado con la beca:

CONACYT 2019-2022

DEDICATORIA A:

A mi madre. Por su esfuerzo en concederme la oportunidad de estudiar y por su constante apoyo a lo largo de mi vida. Por su amor y su confianza en mí desarrollo profesional y académico.

A mis padres. Dagoberto Roblero y Flor González, por darme la vida.

A mi hermana. Helen Roblero, por quererme y cuidar de mí siempre, por todo tu apoyo, tu comprensión en todos estos años de vida y llenarme de inspiración y amor para continuar con esta meta.

A Aurora. Por ser una maravillosa persona tanto en la vida como conmigo, apoyarme siempre incondicionalmente más que nadie en la vida, por ser el pilar más importante de mi vida, brindarme su cariño y darme razones para seguir adelante.

A mis amigos. Gracias por acompañarme en la travesía llamada "vida", por las risas y los malos momentos juntos; y formar parte de la conclusión de este camino junto a mí: Itzcóatl López, Carlos Velasco, Carlos Severino, Carlos Suárez, Julissa Aguilar, Dan Morales y Pablo Sánchez.

A los Doctores y Maestros de la UNAM. Gracias a ellos se pudo llevar a cabo la conclusión de este proyecto.

A la apreciada Universidad Nacional Autónoma de México. Por abrir sus puertas de oportunidades y poder realizar este trabajo dentro de sus instalaciones.

A la Dra. Graciela Velasco. Por continuar apoyando y creyendo en mis capacidades todo este tiempo desde la estancia profesional y durante todo el curso del posgrado, enseñarme y compartir con todos sus conocimientos y brindar todo su apoyo moral.

A mi tía Marisol. Por apoyarme en mi estancia en la ciudad de México, por su cariño y apoyo incondicional.

A mi tío Felipe (RIP). Por todo su apoyo y cariño brindado, ser un buen amigo con el cual contar.

A mi abuela Floriberta (RIP). Por todo su apoyo, amor y consejos que me brindo a lo largo de mi vida, compartir alegrías y conocimientos, y ser una super abuela con quien siempre contar.

A mi padrino Toño Villatoro. Por apoyarme y ayudarme en la etapa más oscura de mi vida, y ser un gran amigo en la vida.

A Diana Laura. Por apoyarme con su amistad, la cual se fue forjando aún más en tiempos de crisis personal, por creer en todo el conocimiento que le confié para el desarrollo de proyectos personales y durante el proceso de desarrollo de esta Tesis, gracias por ser una gran amiga.

A Susan Brigitte. Por llegar en un momento crucial de mi vida, por creer en lo que hago, y darme la confianza para comenzar una nueva etapa de manera conjunta.

Glosario

VGG-16: Modelo de Red Neuronal Pre entrenada

Keras: Librería para programación y manejo de una red neuronal

Kernel: Filtro o matriz por la cual se multiplica una función.

ENSO: El Niño/ Oscilación del Sur

AP: Agricultura de Precisión

IGAC: Instituto Geográfico Agustín Codazzi

DANE: Departamento Administrativo Nacional de Estadística

GPS: Sistema de Posicionamiento Global

GIS: Sistemas de Información Geográfica

ISO: Organización Internacional de Estandarización

SAGARPA: secretaria de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación

SNITT: Sistema Nacional de Investigación y Transferencia Tecnológica para el Desarrollo Rural Sustentable

CSP: Comités Sistema Producto

SINACATRI: Sistema Nacional de Capacitación y Asistencia Técnica Integral

RNA: Redes Neuronales Artificiales

Índice de Figuras

Índice de Tablas

Índice

RESUMEN

En este trabajo se presenta el desarrollo, diseño e implementación de un Sistema de reconocimiento y clasificación de imágenes a través de la creación de una Red Neuronal de datos de plagas y enfermedades de las plantas de cultivo de maíz de riegos. El diseño se ejecuta en programación con lenguaje Python con Spyder (IDE) y algoritmos pre entrenados de modelo VGG-16 y Keras de redes neuronales para el entrenamiento y adaptación de un nuevo banco de imágenes. Este se conforma de 3 etapas, basados en la Agricultura de Precisión: 1) Recolección de Daos, 2) Análisis de Datos y 3) La Implementación. El análisis y diseño del sistema estarán basado en la metodología de Cascada. Este sistema emplea el entorno de desarrollo *Spyder* IDE y VGG-16/Keras, los cuales permiten la compatibilidad con el sistema operativo Windows 10, el lenguaje de programación Python y Google Colab para el diseño y creación del sistema y del banco de imágenes. Este desarrollo es uno de los elementos que integral el *Sistema de Monitoreo Inteligente a través de IA*, que utiliza nuevas metodologías para la detección de imágenes, en particular, para implementar nuevas tecnologías dentro de los centros de riegos de cultivos de la república mexicana para el mejoramiento de los cultivos de maíz.

Capitulo 1

Objetivos

- 1.1 Objetivo General
- 1.2 Objetivos Específicos

1.1 OBJETIVO GENERAL

Desarrollar un sistema de clasificación de imágenes basado en los métodos de Inteligencia Artificial, el cual permita analizar y seleccionar aquellas con enfermedades en las plantas de maíz, en donde se pueda hacer una predicción precisa en los riegos del cultivo de maíz; propiciando a la automatización que beneficie a la Agricultura de Precisión en México, así como apoyar en el desarrollo del sistema que permita mejorar la gestión automatizada del proceso de captura, almacenamiento y análisis de datos de cultivos de maíz..

1.2 OBJETIVOS ESPECIFICOS

- Análisis de Imágenes en Base de Datos.

- Identificar, obtener y recopilar imágenes descriptoras de las plantas de maíz con Mancha Foliar y plantas Sanas, para usarlos como datos de entrada del modelo.

- Realizar evaluación tradicional de la planta de maíz, para poder analizar las metodologías utilizadas para la detección de plagas o enfermedades, las cuales serán usadas como salidas o etiquetas para el modelo.

- Implementación de Keras.

- Creación de Red Convolucional en base al Modelo VGG16

- Entrenar, evaluar y ajustar algoritmos definidos de aprendizaje de la red neuronal con datos almacenados.

- Validar la efectividad del modelo de aprendizaje VGG16 implementando muestras nuevas de maíz no consideradas al principio de la investigación.

Capitulo 2

Justificación

- Justificación

2. JUSTIFICACIÓN

Se determinó viable la realización de sistema de identificación de Enfermedades y Plagas para cultivos de maíz mediante una Red Neuronal, en la cual se integrara un sistema local (*de escritorio*) para el análisis y clasificación de imágenes a partir de una base de datos de imágenes, en particular desarrollar el diseño e implementación de este sistema para la recolección de datos de los diferentes tipos de plagas y enfermedades a través de un banco de fotos que ayuden a la identificación de estos como uno de los nuevos elementos de la cadena de desarrollo tecnológico que tiene la Agricultura de Precisión en México. Este sistema local emplea el entorno de desarrollo Anaconda de Python, dicha plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*, además de la integración de las diferentes librerías que son utilizadas para el desarrollo de las redes neuronales. Un módulo es un archivo desarrollado en lenguaje de programación Python (. Py) que contiene clases escritas en lenguaje Python para interactuar con las API´s y distintos modelos de redes neuronales existentes. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos. Debido a que estos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma Python pueden ser extendidas fácilmente por otros desarrolladores de software. Para este proyecto se contempla albergar información en una carpeta dentro del computador que contenga el banco de imágenes con los diferentes tipos de plagas y enfermedades que se pretenden analizar, para luego, entrenar la red neuronal a partir de estos datos integrados y una vez entrenada, arroje un resultado fiable cada que se pida analizar una nueva imagen.

Capitulo 3

Introducción

- Introducción

3. INTRODUCCIÓN

La agricultura tradicional es altamente vulnerable a diferentes acontecimientos de tipo climático como los fenómenos del niño y niña (ENSO), así como a la presencia de enfermedades y plagas que cada vez son más nocivas y resilientes a agroquímicos tradicionales. Por lo tanto, se plantea la implementación de nuevas tecnologías como alternativas de manejo, monitoreo y control de los cultivos agrícolas de manejo, monitoreo y control de los cultivos agrícolas en diferentes etapas de su desarrollo. Dentro de los registros de los inicios del origen de la agricultura en México datan en los valles de Tehuacán y Oaxaca, se encuentran la domesticación del maíz (ocurrida hace 10,000 a 6250 años aproximadamente) y el hallazgo de que variedades antiguas y actuales del maíz descienden de una misma especie ancestral. La agricultura asegura una producción constante de alimento debido a que las tecnologías asociadas se han innovado constantemente.

A raíz de la actividad intelectual humana en el Medio Oriente y el Mediterráneo para generar conocimientos, se configuro el método científico, basado en supuestos materialistas. Al desentrañar la naturaleza de los fenómenos reales, e incorporar los conocimientos empíricos de todo el mundo, este método permitió que el hombre los manejara. Al aplicarse el método científico occidental a los problemas agrícolas se aceleró la invención y el diseño de maquinaria e instrumentos, así como de formas cada vez más variadas de proveer energía al agro ecosistema en labores como movimiento del suelo, modificación de la topografía, construcción de estructuras para riego y empleo de fertilizantes y combustibles fósiles. En la actualidad la agricultura moderna se caracteriza por la introducción de energía, el uso de productos industriales, mecanización, comunicación y computación electrónica, educación, etcétera.

La agricultura de precisión (AP) es un sistema empleado para analizar y controlar la variación espacio-temporal del terreno y el cultivo. La variación espacial comprende las diferencias en fertilidad de distintas secciones del terreno y las que se dan en el crecimiento de las plantas cultivadas. La AP maneja las variables y administra eficientemente los insumos (por ejemplo, agua o fertilizante). Logra una mayor sostenibilidad al minimizar tanto los recursos invertidos, como el impacto ambiental y los riesgos agroalimentarios, y al mismo tiempo maximiza la

producción. Además, permite reducir hasta un 90% el uso de insumos agrícolas que son liberados al medio ambiente (como pesticidas). Su uso depende de las tecnologías de la información, en donde la comunicación entre dispositivos es una de las herramientas más importantes.

Capitulo 4

Antecedentes

- 4.1 Agricultura Convencional.
- 4.2 Agricultura de Precisión.
- 4.3 Agricultura de Precisión en México.
- 4.4 Tratamiento de Imágenes.
- 4.5 Pre Procesamiento de Imágenes.
- 4.6 Mecanismos de la Visión Humana.
- 4.7 Inteligencia Artificial.
- 4.8 Redes Neuronales.

4.1. Agricultura Convencional

La agricultura es uno de los principales motores económicos del medio rural que, con el paso del tiempo y la aparición de mercados competitivos, requiere de innovación y mejora de la producción de sus productos. En la última década, las variaciones climáticas relacionadas con el fenómeno de El Niño y La Niña, han traído serios retos para la agricultura colombiana, demostrando que muchos agricultores no tienen la capacidad de manejar efectivamente el riesgo ni de adaptarse a fluctuaciones climáticas y catástrofes. Al mismo tiempo, la mayoría de modelos de cambio climático predice que los daños serán compartidos de forma desproporcionada por los pequeños agricultores del tercer mundo, lo que aumenta su vulnerabilidad.

Según cifras del Instituto Geográfico Agustín Codazzi (IGAC), en el 2013, de los 22,1 millones de hectáreas para uso agrícola con las que cuenta el país gracias a su condición tropical y variedad de pisos térmicos, Colombia utiliza 5,3 millones, es decir, solo el 24,1% de sus capacidad y potencial agropecuario. De acuerdo con Perfetti (2013), cifras del Departamento Administrativo Nacional de Estadística (DANE) indican que, en 2012, 11,204,685 personas habitaban zonas rurales del país y tenían su sustento en la actividad agropecuaria. Asimismo, de acuerdo con Fajardo-Junco (2013), la mayoría de estos son pequeños productores con prácticas de trabajos tradicionales que implican una baja tecnificación en el desarrollo de los cultivos. De un lado, la disposición de los recursos naturales, tales como, agua, aire y suelo ha disminuido en calidad y cantidad; mientras que la variabilidad climática, como componente horizontal de todas las actividades desarrolladas alrededor del agro, ha aumentado y generado pérdidas económicas de grandes y pequeños productores agrícolas. En ese panorama, la escasa planificación de los cultivos de gran parte del sector agrícola, y de los sobrecostos asociados a la atención de enfermedades, así como la baja asistencia técnica y la deficiente implementación de nuevas tecnologías en cada una de las fases de desarrollo del sistema productivo, desencadenan efectos que se traducen en una baja competitividad del agricultor y poca inserción del producto en mercados que exigen estándares de calidad relacionados con menor uso de agroquímicos, fertilizantes y plaguicidas, así como su aplicación directa, no solo sobre las áreas que lo requieren, sino sobre toda la extensión del cultivo, promueve el aumento de la resilencia de plagas y enfermedades de los cultivos que cada vez demandan un manejo más invasivo, genera

mayores inversiones económicas para combatirlas, afecta la calidad de los componentes nutricionales del producto final e influye en la disminución en la calidad de vida de las familias campesinas, en su derecho a la vida y a la alimentación. Por su lado, Fajardo-Junco (2013), sostiene que el uso de herramientas tecnológicas como las imágenes de satélite propuestas en la agricultura de precisión, ha presentado inconvenientes relacionados con el tiempo, el costo y la calidad de las imágenes en cuanto a su resolución espacial, mientras que los estudios realizados en Colombia alrededor del uso de aviones no tripulados han estado, por lo general, enfocados a misiones institucionales y aplicaciones militares.

Países como Estados Unidos, España y Brasil, han avanzado en su implementación para la planificación y manejo de grandes extensiones de cultivos; por ejemplo, en el mapeo de brotes de hierbas invasoras, riego necesario, zonas de fertilización y anomalías presentes en cultivos de fruta o en la identificación de la participación de la luz y la transferencia de energía, el enfriamiento y la foto protección en la cosecha de vid mediante el uso de imágenes híper espectrales, que permiten reconocer las propiedades de la superficie terrestre y la estimación de propiedades de la superficie terrestre y la estimación de propiedades geo biofísicas a partir de la interacción de la radiación electromagnética con el material del suelo o las plantas.

4.2Agricultura de Precisión

Es un sistema empleado para analizar y controlar la variación espacio-temporal del terreno y el cultivo. La variación espacial comprende las diferencias en fertilidad de distintas secciones del terreno y las que se dan en el crecimiento de las plantas cultivadas. La variación temporal engloba las diferencias observadas en la producción de un mismo terreno entre una temporada y otra (Tabla 1). Aun en terrenos con poca extensión, de una hectárea o menos, existe dicha variación.

La AP maneja las variables y administra eficientemente los insumos (por ejemplo, agua o fertilizantes). Logra una mayor sostenibilidad al minimizar tanto los recursos invertidos, como el impacto ambiental y los riesgos agroalimentarios, y al mismo maximiza la producción. Además, permite reducir hasta el 90% el uso de insumos agrícolas que son liberados al medio ambiente (como los pesticidas). Su uso depende de las tecnologías de la

información, en donde la comunicación entre dispositivos es una de las herramientas más importantes.

La AP no consiste solamente en medir la variabilidad existente en el área, sino también en la adopción de prácticas administrativas que se realizan en función de esa variabilidad. De acuerdo con Robert (1999), la observación de la existencia de variabilidad en las propiedades o factores determinantes de la producción en los agro ecosistemas no es una novedad. Lo que es diferente, en realidad, es la posibilidad de identificar, cuantificar y mapear esa variabilidad. Más aun, es posible georreferenciar y aplicar los insumos con dosis variables en puntos o áreas de coordenadas geográficas conocidas.

Tipo de variación	Sujeto de la variación	Factores involucrados
Espacial	Fertilidad del suelo	• Condiciones fisicoquímicas (entre otras la acidez-alcalinidad o pH, y el contenido de nitrógeno o de metales). • Contenido de humedad, materia orgánica y contaminantes. • Conductividad eléctrica e hidráulica. • Textura, fuerza mecánica y profundidad. • Salinidad. • El relieve o topografía del terreno. • Microbiota y fauna del suelo.
	Desarrollo vegetal	• Maleza (plantas oportunistas). • Plagas (insectos, virus y microorganismos). • Características genéticas del cultivo (como la resistencia a la sequía y velocidad de desarrollo).
Temporal	Cosecha	• Variación productiva entre períodos de siembra distintos. • Condiciones climatológicas (por ej. radiación solar o humedad ambiental) entre distintas temporadas.

Tabla I. Variación espacio-temporal en cultivos (INCyTU, 2018)**.**

Diferencia entre la Agricultura de Precisión y la convencional

La agricultura convencional considera que un terreno es homogéneo y aplica los insumos con base en valores promedio a toda la superficie de siembra; esto incrementa los costos de inversión y el impacto ambiental (como la contaminación del subsuelo). En contraste, en la AP se aplican distintas cantidades de insumos y se valoran las necesidades particulares de cada sección de cultivo y su respuesta en tiempo real (Recuadro I).

- **Agricultura convencional.** Considera a las condiciones de siembra como homogéneas.
- **Agricultura de precisión.** Maneja la variabilidad espacio-temporal, maximiza el rendimiento y reduce costos de inversión e impacto ambiental.

Recuadro I. Diferencias entre la Agricultura Convencional y de precisión (INCyTU, 2018)**.**

Etapas de la Agricultura de Precisión

Se requiere de tres etapas (Fig. I):

1. *La recolección de datos.* Se lleva a cabo con equipos especializados como satélites o sensores remotos.
2. *El análisis de los datos.* Un experto analiza los datos y emite sugerencias para manejar adecuadamente la variación espacio-temporal detectada.
3. *La implementación.* El productor cultiva el terreno según las recomendaciones

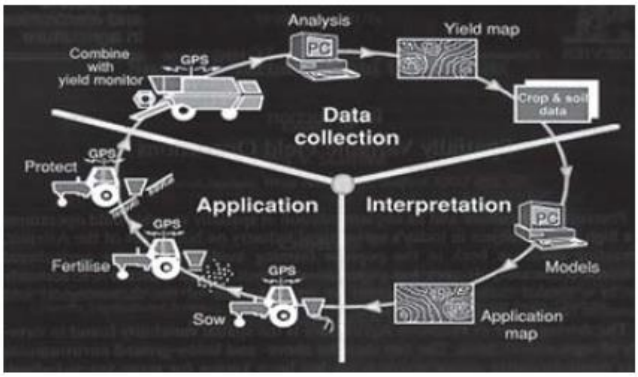


Fig. I Etapas de la Agricultura de Precisión (Rodolfo Bongiovanni, 2006)

Tecnologías Asociadas

Existen cinco tecnologías asociadas a la AP: los sistemas de posicionamiento global (GPS por sus siglas en ingles) y de información geográfica (GIS), sensores remotos, monitores de rendimiento/aplicación y maquinaria inteligente. Ya existen empresas que brindan servicios especializados en estas tecnologías.

1.- Sistemas de posicionamiento global

El GPS fue desarrollado por el ejército de los Estados Unidos para brindar servicios de posicionamiento y navegación global. Lo constituyen tres segmentos: espacial, de control y

de usuario. El segmento espacial lo conforma una constelación de satélites. El segmento de control está formado por estaciones ubicadas en distintos puntos del planeta. El segmento de usuario lo representan los equipos receptores de la señal satelital.

Dado que el GPS convencional tiene un margen limitado de precisión, se introdujo el GPS diferencial (GGPS, por sus siglas en ingles), para corregir errores durante la asignación de coordenadas. Esta corrección usa una estación receptora fija que compara sus coordenadas con aquellas obtenidas por el satélite. En la AP el DGSP ha permitido registrar la variabilidad espacio-temporal y controlar con exactitud geográfica la maquinaria agrícola. Este se emplea en monitores de rendimiento, banderilleros satelitales, pilotos automáticos o en equipos de aplicación variable.

2.- Sistemas de información geográfica

Los GIS son sistemas informáticos usados para almacenar, visualizar y analizar datos referidos geográficamente. En la AP permiten analizar la información obtenida mediante distintos receptores (v.g. sensores remotos), para tomar decisiones sobre el manejo de la variabilidad espacio-temporal. Existen varios GIS disponibles comercialmente. También se emplean programar informáticos para interconectar los dispositivos con computadoras personales.

El reto con esta tecnología es analizar los datos adecuadamente. El software agrícola aún no posee un consenso internacional en lo referente a la comunicación entre las distintas tecnologías empleadas en AP, aunque existen iniciativas de estandarización (propuesta por la Organización Internacional de Estandarización, ISO por sus siglas en inglés, en su norma ISO11783). La estandarización permitirá la compatibilidad entre distintos dispositivos.

3.- Sensores Remotos

Los sensores remotos son sistemas (satelitales o portátiles) que obtienen información del cultivo, sin tener un contacto físico con este. Se emplean en la recolección de datos sobre la administración del agua de riego, contenido de materia orgánica, vigor de las plantas (por ejemplo, su contenido de clorofila), enfermedades vegetales, plagas, mapeo de malezas, sequia e inundaciones. Para ser eficientes, deben estar bien calibrados y poseer suficiente resolución.

Otra forma de percepción remota es la fotografía, que puede obtenerse vía satelital o área por medio de aviones o drones. La resolución de estas imágenes depende de las capacidades del equipo utilizado.

4.- Monitores de rendimiento y aplicación

Los monitores de rendimiento obtienen información sobre la cantidad (granos recolectados por unidad de tiempo) y la calidad del cultivo (entre otras características, la humedad del producto). Otro tipo de monitores, los de aplicación variable, se usan para dosificar la cantidad de insumos por cada sección del terreno, por ejemplo, la dosis de semilla o agroquímicos. Ambos tipos de monitores dependen del DGPS.

Con la información reunida se crean mapas de productividad y de características del suelo, así como modelos de crecimiento vegetal. Con estos mapas, se aplican los insumos (como herbicidas) cubriendo las necesidades particulares de cada zona del cultivo.

5.- Maquinaria inteligente

La cosecha de algunos productos, por ejemplo, frutas, requiere una intensa labor manual por parte de trabajadores temporales, lo que incrementa los gatos de producción. Se han desarrollado cosechadoras inteligentes capaces de diferenciar frutos maduros e inmaduros. También existen sistemas de detección de flores en árboles frutales, que permiten estimar la producción, la aplicación variable de agroquímicos y la recolección automatizada de frutos. Además, existen sistemas de piloto automático que controlan la maquinaria agrícola vía DGPS. La instigación básica y aplicada en inteligencia artificial impactara positivamente a la AP.

Tecnología	Ejemplos
GPS	• Satélites NAVSTAR
GIS y software relacionado	• ARC-INFO • Agri-Logic • Magellan Waypoint
Sensores remotos	• Sensores de nitrógeno • Agro Drone • Satélites Landsat
Monitores de rendimiento y aplicación	• AG Leader • Green Star John Deere
Maquinaria inteligente	• Detección/recolección de frutos • Piloto automático en tractores • Uso de inteligencia artificial

Tabla II. Principales Tecnologías de la AP (INCyTU, 2018)**.**

Beneficios Potenciales.

La adopción de la agricultura de precisión, no solamente como utilización de tecnologías de información, sino como concepto, es un potencial para la racionalización del sistema de producción agrícola moderno como consecuencia de:

- Optimización de la cantidad de agroquímicos aplicados en los suelos y cultivos;
- Consecuente reducción de los costos de producción y de la contaminación ambiental; y
- Mejora de la calidad de las cosechas.

Si bien es un tema de investigación relativamente nuevo, se han logrado muchos avances, principalmente en el desarrollo de máquinas e implementos que permiten el manejo localizado en base a mapas. Los recursos más avanzados en tecnología de información hoy disponibles, como los sistemas de posicionamiento global (GPS), los sistemas de información geográfica (SIG), los sistemas de control y adquisición de datos, sensores y actuadores, entre otros están cada vez más presentes en el campo. A pesar de ese avance tecnológico, hay áreas que necesitan desarrollarse aún más para que la agricultura de precisión pueda consolidarse como una solución amplia y plenamente viable, para todos los segmentos de la agricultura. Se destacan dos grandes áreas de trabajo:

1.- El desarrollo de sensores que permitan obtener, en tiempo real, de forma eficiente y confiable, la deficiencia nutricional o de estrés hídrico de la planta durante su desarrollo para aplicación y corrección en el tiempo preciso y;

2.- El desarrollo de dispositivos, programas de computación y estrategias que posibiliten una mayor integración de los datos obtenidos, facilitando así la interpretación y análisis de los mapas y haciendo también más efectivo el manejo localizado.

Actualmente, uno de los más grandes problemas que se plantea es la magnitud de la correlación de la variabilidad espacial y/o temporal entre los factores asociados al suelo y al desarrollo de los cultivos, incluyendo disponibilidad de nutrientes, materia orgánica, acidez, disponibilidad de agua, textura, distribución de enfermedades, plagas, malezas, etc. La determinación de esos factores es de suma importancia para poder distribuir las cantidades ideales de agroquímicos, fertilizantes o correctivos. Una de las técnicas que viene llamando la atención de la investigación mundial recientemente, es el procesamiento y análisis de

imágenes satelitales, aéreas o hasta de plataformas instaladas en las máquinas agrícolas (Figuras II Y III).

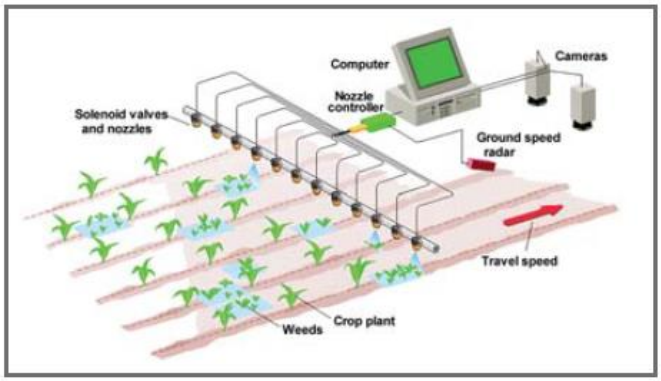


Fig. II Esquema de un pulverizador para control localizado de malezas basado en el procesamiento y análisis de imágenes (Rodolfo Bongiovanni, 2006).

.

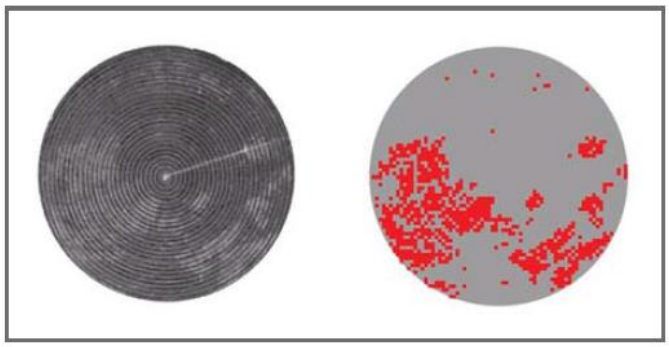


Fig. III Mapa de infestación con malezas en una plantación de café a la derecha, desarrollada en base al procesamiento y análisis de la imagen aérea de la izquierda (Rodolfo Bongiovanni, 2006).

4.3 Agricultura de Precisión en México

En México el organismo encargado de regular y promover al sector agroalimentario es la secretaria de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación (SAGARPA). A pesar de que el Reglamento Interior de la SAGARPA no menciona la AP, los artículos 18 y 19 permiten su aplicación en el sistema agropecuario nacional. De igual manera, el Congreso de la Unión ha impulsado el desarrollo sostenible de las actividades agropecuarias en la Ley de Desarrollo Rural Sustentable.

Entre los datos más importantes relacionados con la actividad agropecuaria mexicana, destaca:

- Aproximadamente 24% de la población total habita zonas rurales.
- Cerca de 4% del producto interno bruto corresponde a la agricultura.
- Existe baja productividad agrícola.
- La SAGARPA y el sector privado ofrecen asistencia técnica.
- Los centros de investigación y universidades no cuentan con modelos de transferencia tecnológica hacia el campo.
- El sistema de innovación mexicano carece de interacción-colaboración institucional.
- La investigación agrícola se realiza principalmente con recursos públicos (Fundaciones Produce, SAGARPA o CONACYT).
- Se crearon el Sistema Nacional de Investigación y Transferencia Tecnológica para el Desarrollo Rural Sustentable (SNITT, 2004), Comités Sistema Producto (CSP, 2003) y el Sistema Nacional de Capacitación y Asistencia Técnica Integral (SINACATRI, 2003).

En México el uso de la AP podría reportar beneficios económicos (para agricultores, industria y gobierno), sociales (para asegurar la disponibilidad de alimento suficiente) y ambientales (para reducir el riesgo de contaminación por agroquímicos), como los observados en países que la han adoptado.

La estrecha relación que guarda con la tecnología podría comprometer algunos empleos en zonas rurales, por lo que será necesario diseñar estrategias que maximicen los beneficios para todos los sectores involucrados.

Algunas acciones que ayudarían a fomentar su adopción son la colaboración entre academia, industria, gobierno y los productores; la actualización de la Ley de Desarrollo Rural Sustentable y del Reglamento de la SAGARPA; la investigación y desarrollo agrícola en México; la formación de recursos especializados (humanos e institucionales); la promoción de programas piloto para cultivos estratégicos (como el aguacate) y su divulgación.

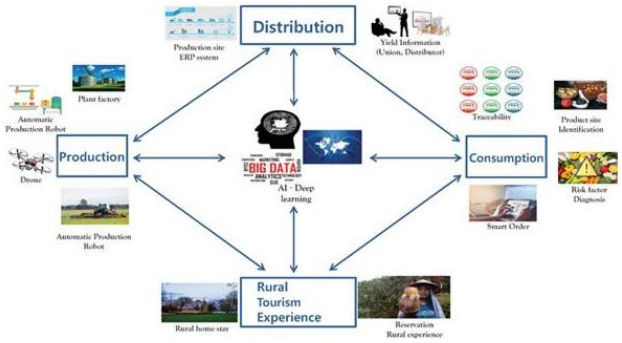


Fig. IV Representación de los diferentes métodos de aplicación de la AP

4.4 Tratamiento de Imágenes

En este capítulo se abordan algunos de los conceptos necesarios para justificar el empleo de las redes neuronales como clasificadores no paramétricos en la clasificación de imágenes digitales en la agricultura de precisión. Se habla del proceso de filtrado como herramienta computacional para reducir el efecto del ruido causado en una imagen. Se expone dos filtros utilizados para esta investigación: Filtro Gaussiano y Filtrado Anisotrópico.

Hoy en día, los robots se utilizan en múltiples situaciones: en labores de rescate, de manipulación de residuos sólidos, en la enseñanza, etcétera. Para que dichas actividades puedan ser ejecutadas, dichas maquinas se valen de sensores, por ejemplo, la cámara digital; a través de este dispositivo, el robot adquiere imágenes o video del entorno circundante. Dichas imágenes deben ser tratadas y analizadas para que el robot sea capaz de ubicar e identificar los objetos a su alrededor y así desplazarse entre ellos y eventualmente para realizar tareas específicas con los mismos, como manipuladores. Para esto, el sistema de cómputo del robot debe realizar sobre las imágenes adquiridas un conjunto de pasos estratégicamente escogidos, con el objeto de obtener el mejor desempeño posible (Fig. V).

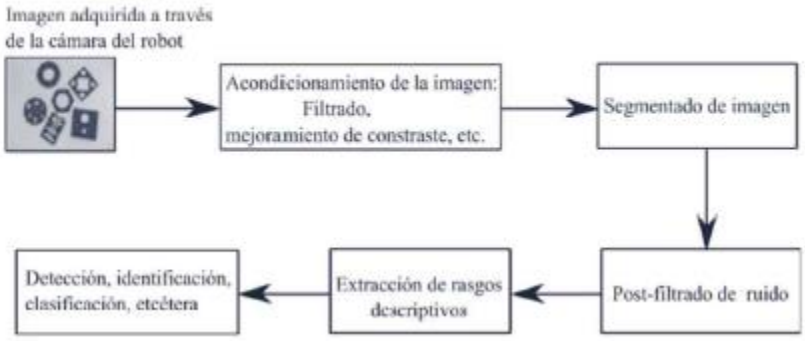


Fig. V. Secuencia de pasos realizados sobre una imagen a la entrada del sensor del robot para que este último sea capaz de ubicarse, desplazarse y realizar tareas con los objetos a su alrededor dentro de su espacio de trabajo (Humberto S. Azuela, 2020)**.**

Una vez que la imagen de entrada ha sido tratada en forma digital, esta debe ser segmentada, es decir, dividida en regiones con un significado lo más cercano al de los diferentes objetos en la escena observada por la cámara. Las regiones segmentadas todavía pueden venir acompañadas de ruido parásito. Esto se puede lograr mediante técnicas de post- filtrado, enseguida, para cada uno de los objetos segmentados de la imagen se calcula un conjunto de rasgos de tipo geométrico o topológico que permiten describirlos en forma global o local. Estas descripciones pueden ser usadas en una etapa final para detectar, identificar y ubicar dentro de la imagen y, en consecuencia, dentro del espacio de trabajo del robot a los diferentes objetos.

4.4.1. Función Imagen

Una función de imagen o imagen es cualquier función real *g (x, y)* con integral finita y soporte compacto *SC* tal que, para todo punto $p \in SC, x > 0, y > 0$.

Para ser útil, una imagen de acuerdo con la definición anterior, requiere ser muestreada y cuantizada. Estas operaciones se realizan normalmente a través de lo que se conoce como digitalizador o numerizador. El resultado de digitalizar una imagen, da como resultado una imagen digital.

4.4.2 Imágenes Digitales

Una imagen natural captada con una cámara, un telescopio, un microscopio o cualquier tipo de instrumento óptico presenta una variación de sombras y tonos continua. Imágenes con estas características se denominan *imágenes analógicas.*

Para que una imagen natural o analógica pueda ser "manipulada" por un medio de cómputo, primero debe ser digitalizada. (Bustos *et al.,*2004).

Denotada como *f (x, y)* es un arreglo bidimensional $\in Z \times Z$, si *(x, y)* son números enteros de $Z \times Z$ y *f* una función que asigna a cada par *(x, y)* un numero de *Z*; donde *Z* es el conjunto de los enteros.

Fig. VI. Tres imágenes digitales muestreadas a una resolución de 416x312 pixeles y cuantizadas a 256 niveles de gris (Humberto S. Azuela, 2020)**.**

4.4.3 Imagen Binaria

Una imagen binaria denotada como *b (x, y)* es una imagen digital *f (x, y)* que ha sido cuantizada a dos niveles de intensidad, 0 y 1.

Fig. VII. Versiones binarias de las tres imágenes digitales de la Fig. VI (Humberto S. Azuela, 2020)**.**

Se muestran tres versiones binarias de las imágenes digitales mostradas anteriormente. Nótese que a pesar de que la información de las imágenes originales fue reducida de 256 a dos niveles de gris, la información esencial de los objetos contenidos se sigue manteniendo. Esto, es esencial en tareas como la detección y la identificación de objetos.

4.4.4 Representación

Una imagen se representa por una función bidimensional *f (x, y)*, cuyo valor corresponde a la intensidad de luz en cada punto del espacio de las coordenadas *(x, y)*. en el caso de una imagen monocromática, al valor de la función se le denominara *nivel o escala de gris* en el punto de coordenadas *(x, y).* Las imágenes a color están formadas por la combinación de imágenes 2-D. En base a este concepto, una imagen es analógica si el dominio (valores de (x, y)) y el rango (valores de *f (x, y))* son continuos, mientras que una imagen digital posee dominio y rangos discretos.

Para convertir una imagen analógica en digital, la imagen es dividida en valores de brillo individuales, mediante dos procesos denominados *muestreo (sampling)* y *cuantización (quantization)* (Mihaich, 2014).

4.4.5 Resolución espacial, resolución espectral y profundidad de color

Las dos fundamentales causas de pérdidas de información cuando se captura una imagen digital son la naturaleza discreta del valor de los pixeles y el rango limitado de los valores de intensidad luminosa que pueden tener cada uno de estos elementos. En base a estos problemas, surgen los conceptos de *resolución espacial* y profundidad *de color.*

Resolución Espacial:

El muestreo determina la resolución espacial de la imagen. La resolución espacial de una imagen es una característica de la misma determinada por las características del sensor y las condiciones de adquisición de la imagen (Recio, 2010). Define el menor detalle discernible dentro de la imagen, es decir, el menor número de pixeles comprendidos en una unidad de distancia (por ejemplo, 500 pixeles por centímetro) (Mihaich,2014).

La resolución espacial suele interpretarse como el tamaño del objeto más pequeño que puede ser distinguido en una imagen: tamaño del pixel sobre el terreno (Pérez Gutiérrez and Muñoz Nieto, 2006).

Un pixel no representa un solo punto en la imagen, sino una región rectangular. Con pixeles grandes no solo se pierde en resolución espacial, sino que el valor del nivel de gris

correspondiente puede hacer aparecer discontinuidades o fronteras en los bordes de los pixeles. En dichos casos es necesario realizar una clasificación subpíxel para obtener un grado de detalle superior al de la imagen. En cambio, cuanto menor sea el tamaño del pixel sobre el terreno menor será la posibilidad de que se encuentren dos o más fronteras dentro de la imagen dentro de él.

Por lo tanto, dependiendo de lo que se quiera tratar en la imagen, un tamaño de pixel será más funcional que otro. En el caso de que se requiera disminuir la variabilidad de categorías dentro de la imagen, una menor resolución espacial es la técnica empleada, mientras que, en el caso que se quieran tratar imágenes con una alta precisión, se emplea una resolución espacial mayor. A medida que los pixeles se hacen más pequeños, ocurre la sensación de que la imagen es continua. Esto pasa porque el tamaño de los pixeles es menor que la resolución espacial de nuestro sistema visual.

Fig. VIII Imagen representada en diferentes resoluciones espaciales (Regalado, 2018)**.**

Resolución espectral:

Esta propiedad viene determinada por el número de bandas del espectro electromagnético que es capaz de discriminar el sensor. Hay sensores que captan la información en las bandas del espectro visible. Sin embargo, existen sensores capaces de recoger información de longitudes de onda invisibles al ojo humano, por ejemplo, el infrarrojo cercano, cuyas longitudes de onda son algo menores que la menor longitud de onda que puede detectar el ojo humano (Pérez and Muñoz, 2006).

Los nuevos sensores, llamados también espectrómetros o hiper espectrales llegan a tener hasta 256 canales con un ancho de banda muy estrecho (unos pocos nm) para poder separar de forma precisa distintos objetos por su comportamiento espectral.

Así que, básicamente, la resolución espectral difiere en la habilidad de los sistemas de percepción de distinguir y diferenciar entre el espectro de radiación electromagnética de distintas longitudes de onda (García-Cervigón, 2015). En cuanto a resoluciones espectrales se pueden distinguir entre imágenes:

- Multiespectrales: Que generalmente capturan información entre 3 y 7 bandas de unos 100 nm de ancho.
- Hiper espectrales: Que adquieren información en varias decenas o centenas de bandas, hasta 256 generalmente, con longitudes de onda inferiores a los 5 nm de ancho, permitiendo así separar de forma muy precisa distintos objetos por su comportamiento espectral.

Profundidad de color

La cuantización viene a resolver la imposibilidad de tener un rango infinito de valores para la intensidad o brillo de los pixeles. Después de que se captura la imagen, se le asigna a cada pixel una intensidad representada por un número entero. La apreciación de este valor es directamente proporcional al número de bits que utiliza el dispositivo con que se captura la imagen para representar los números enteros (Mihaich, 2014).

Por tanto, la profundidad de color representa el número de bits necesarios para la codificación y el almacenamiento de la información de color de cada pixel presente en la imagen. Un bit es una posición de memoria que puede tener el valor 0 o 1. Así, mientras mayor sea la profundidad de color en bits, mayor será la paleta de colores presente dentro de la imagen. Si se emplea un bit, la imagen será en blanco y negro (0= color negro y 1= color blanco), mientras que, si se utilizan 8 bits, la imagen adquirirá 256 niveles de gris.

4.4.6 Modelos de color

Un modelo de color es un modelo matemático abstracto que describe la forma en que los colores pueden ser representados de forma numérica. Tienen como objetivo facilitar la especificación de los colores de forma normalizada y aceptada genéricamente (Mihaich,2014).

Entre los modelos de color más empleados en el procesamiento de imágenes están: el modelo RGB y el modelo CMY.

El modelo *RGB* forma cada color de la imagen como la combinación de tres canales correspondientes a los colores primarios: rojo *(Red),* verde *(Green)* y azul *(Blue)*. Es un modelo de color basado en la síntesis aditiva: cada color se representa como la suma de los colores primarios, siendo el blanco la suma de todos ellos (Baluja, 2009).

Este modelo no define por sí mismo lo que significa exactamente rojo, verde o azul; por lo que los mismo valores RGB pueden mostrar tonos notablemente diferentes en dispositivos diferentes.

El modelo *CMY* es el inverso del modelo RGB: en este caso el origen es el blanco y los ejes primarios son los colores cian *(Cian)*, magenta *(Magenta)* y amarillo *(Yellow)*. Este modelo es sustractivo, la suma de todos los colores produce el negro (Mihaich, 2014).

Si se muestra una imagen CMY como si fuera RGB se podrá observar una imagen con todos sus colores invertidos o negativos.

Las ecuaciones (1-6) permiten pasar de un sistema a otro:

c= max - r (1) m= max - g (2) y= max - b (3)

r= max - c (4) g= max - m (5) b= max – y (6)

Donde:

- *Máx. es el valor de la intensidad, c, m e y son las componentes C, M e Y respectivamente del modelo CMY, y r, g y b son las respectivas componentes R, G y B del modelo RGB.*

4.4.7 Propiedades de la imagen

Las imágenes presentan un conjunto de características descriptivas inherentes. Entre ellas están: el tono, el color, la textura, la forma, el tamaño y el patrón (Vargas, 2008).

Tono:

El tono describe el brillo relativo de los objetos. Uno de los principales criterios de interpretación visual de una imagen es la variación en el tono de esta. La cantidad de energía reflejada por la superficie de un objeto está directamente relacionada con la expresión del

tono del mismo. Las diferentes clases de objetos presentan variaciones en el tono entre ellas, y a su vez los objetos que pertenecen a una misma clase también difieren en el tono entre ellos. Por ejemplo, el suelo y la vegetación presentan diferencias notables en tono, pero, un suelo desértico y un suelo con alto contenido de óxidos ferrosos difieren en cuanto al tono.
Diversos factores inciden en la representación del tono en una imagen, entre ellos:

- La posición del sol (su altura en el cielo y la estación del año).
- La distinta reflectividad en la banda analizada (por ejemplo, en el espectro visible la vegetación presenta tonos oscuros, mientras que en las longitudes de onda del infrarrojo presentan tonos más claros).
- Las características de los objetos dependiendo de la época del año (por ejemplo, un rio en la época húmeda presenta un ensanchamiento, presentando ambos diferentes tonos).

Se destaca el hecho de que, al representar una imagen en formato digital, se pierden tonos, puesto que los detectores, en su mayoría, detectan 256 niveles de gris. Por otra parte, el ojo humano no está capacitado para distinguir 256 niveles de gris.

Color:

El ojo humano esta apto para percibir longitudes de onda entre los 350 y 780 nm, separando la energía recibida en tres componentes que dan su nombre a los colores primarios: rojo, verde y azul. En la retina hay células foto sensitivas denominadas conos. Hay tanta cantidad de tipos de conos como colores primarios. Por esta razón se dice que la visión humana es tricromática (Pinto, 2006).

Cualquier combinación entre los colores primarios genera un nuevo color, y a su vez, cada color puede ser representado como una mezcla entre ellos.

Los dispositivos de visualización digitales tienen tres canales: azul, rojo y verde. En el caso que se desee visualizar una sola banda del espectro, por ejemplo, grises, se introduce la misma información por los otros tres canales del dispositivo. En el caso de que sea una composición de color uno de los canales, de forma que se reproducen multitud de colores como producto de la combinación de los valores de intensidad de cada una de las tres bandas por cada pixel.

Textura:

La textura es la frecuencia con la que se suceden cambios tonales, es decir, la forma en que se contrastan especialmente los elementos que componen la imagen. Esta característica se produce como una combinación de rasgos unitarios que pueden ser demasiado pequeños para diferenciarlos individualmente, pero que juntos marcan una diferencia respecto al resto de la foto.

En la vegetación, por ejemplo, cada hoja tiene su propia forma, tamaño, patrón, sombra y tono, pero todas estas características junta hacen que sea posible diferenciar entre un tipo de vegetación y otra. La textura está estrechamente ligada a la resolución espacial del sensor, ya que procede de la relación entre el tamaño del objeto y dicha resolución. A medida que aumenta la altura a la que se fotografía el objeto o área, la textura se hace progresivamente más fina hasta desaparecer (Arista *et al.* 2017).

Tamaño:

El tamaño de los objetos se tiene que determinar en el contexto de la resolución espacial y la escala a la que se muestra la imagen. También es importante relacionar el tamaño del objeto analizado con otros objetos de la imagen, para saber, por ejemplo, si una carretera o camino es más o menos importante.

Patrón:

se refiere a la distribución espacial de los objetos en ciertas formas cada cierta área. Se habla de patrón concéntrico, de patrón rayado, de patrón radial, de patrón cuadriculado, etc.

4.4.8 Proceso de reconocimiento y clasificación de imágenes

Un sistema de visión por computadora y la posterior clasificación de las imágenes obtenidas permiten realizar la identificación y clasificación de los objetos contenidos dentro de las imágenes siguiendo un procedimiento que incluye (Hernández, 2014):

- Mejoramiento de la imagen
- Segmentación y etiquetado

- Representación y descripción
- Reconocimiento de formas

4.4.8.1 Mejoramiento de la imagen

El mejoramiento de la imagen se centra en la eliminación de ruido en la imagen, realce de bordes, elección de los mejores de brillo y contraste, eliminación de los efectos de distorsión introducidos por el mecanismo de captura de la imagen, etc.

Un ejemplo claro es cuando se incrementa el contraste de una imagen debido a que "se ve de una mejor forma".

Estas "mejoras" generalmente se consiguen operando con el histograma de la imagen, el cual es una representación del nivel de intensidad de cada pixel por la cantidad de veces que aparece en la imagen. En el caso de que se quiera representar esa escala en referencia a la unidad, se divide cada valor por la cantidad de pixeles de la imagen, obteniéndose así un histograma normalizado.

Una de las operaciones básicas que se realiza sobre el histograma es la ecualización, la cual consiste en balancear la frecuencia de los niveles de intensidad de una imagen, dando como resultado una imagen con mejor contraste.

4.4.8.2 Representación y descripción

La representación y descripción casi siempre es la etapa siguiente a la segmentación y etiquetado, cuya salida es usualmente un conjunto de pixeles los cuales constituyen a un objeto en la imagen o región.

La primera decisión que se debe tomar es si el conjunto de datos representa los límites de una región o la región completa. La representación de limites es apropiada cuando el problema se enfoca en las características extremas de un objeto, como lo son las esquinas o puntos de inflexión. La representación de las regiones es útil cuando se orienta a las propiedades internas de los objetos, como son la textura o la forma del esqueleto de los objetos (Hernández, 2014).

4.4.8.3 Reconocimiento de formas

El reconocimiento de formas es el proceso que se le asigna a un nombre “con un significado” a un objeto, el cual se basa en su descripción y características. El reconocimiento de un objeto requiere un conocimiento previo de los que son los objetos. A ese conocimiento se le denomina “base de conocimiento” (Mihaich,2014).

4.4.9 Clasificadores de imágenes

Un clasificador de imágenes es un método que procesa una imagen o un conjunto de ellas y retorna la imagen clasificada en los objetos que la componen. Existen varios tipos de clasificadores:

4.4.9.1 Métodos de clasificación según la unidad espacial

La resolución espacial de una imagen es una característica determinada por las características del sensor y por las condiciones de adquisición de la imagen. Siendo una de las características de la imagen, el pixel no debe de condicionar totalmente la metodología a utilizar ni el tamaño de los objetos a extraer del análisis de la imagen.

En determinados problemas, el tamaño del pixel será demasiado grande en relación con el tamaño de los objetos a identificar, requiriendo de una clasificación sub pixel para obtener un grado de detalle superior en la imagen. En cambio, en el caso de imágenes de alta resolución, se hace necesario el análisis de una región de pixeles con propiedades similares (Recio,2010).

A continuación, se realiza un compendio de los principales métodos de clasificación y su proyección en la agricultura de precisión.

Clasificadores por pixel:

Los clasificadores por pixeles tradicionales obtienen un vector de características para cada clase a partir de las propiedades espectrales de todos los pixeles contenidos en sus respectivas áreas de entrenamiento.

Cada pixel es asignado a una categoría que es exclusiva. Esta metodología da buenos resultados en las cubiertas espectralmente homogéneas, pero proporciona resultados menos satisfactorios en las cubiertas heterogéneas. Sus principales inconvenientes son que no consideran la información espacial de la imagen y la existencia de pixeles mixtos, o de borde, que representan mezclas de dos o más tipos de clases (Blaschke, Burnett and Pekkarinen, 2004).

Aunque la clasificación por pixel se ha demostrado útil para la clasificación de cultivos, hay dos problemas de clasificación frecuentes: la variabilidad espectral de una cubierta dentro de una parcela agrícola debido a, por ejemplo, variaciones de la humedad del suelo, diferencias en nutrientes, enfermedades, etc. Esta heterogeneidad espectral en el interior de una parcela permita que sucedan clasificaciones erróneas, aun teniendo toda ella el mismo cultivo. Por otro lado, se presenta la aparición de pixeles mixtos situados en la zona de contacto de dos parcelas limítrofes. En algunos casos, la huella espectral de los pixeles mixtos, es más similar a un cultivo distinto a los dos presentes en esa superficie, con lo que su clasificación será incorrecta (Smith and Fuller, 2004).

Clasificadores subpíxel:

La suposición de que cada pixel pertenece a una clase informacional no es correcta en todos los casos, especialmente cuando se trata de paisajes complejos y la resolución empleada es media o baja. La existencia de pixeles mixtos dificulta la aplicación de los algoritmos clasificadores por pixel. Por tanto, se hace necesario un enfoque diferente.

En lugar de un clasificador "duro", entendido como el que asigna una única clase a cada pixel, se utilizan clasificadores blandos que determinan el grado de pertenencia de un pixel a cada clase. Entre estos métodos destaca la clasificación difusa o borrosa (*fuzzy*), que determina para cada pixel el grado de pertenencia a las distintas clases a partir de las funciones de pertenencia definidas para cada vez de ellas (Recio, 2010).

Clasificadores de contexto:

En la clasificación por pixel se utiliza el nivel digital del pixel en varias bandas espectrales, es decir, los valores que le corresponden a un pixel en un espacio multidimensional de características, pero no se considera el contexto espacial en que se encuentra. El contexto

espacial ha tenido una importancia reducida en el análisis de imágenes basado en pixeles. Se ha centrado la atención en el análisis estadístico de los valores almacenados en los pixeles, más que en descubrir y entender los patrones espaciales que ellos siguen.

Entre los rasgos más importantes para la interpretación visual humana siempre se incluyen las características espaciales de la imagen como son la textura, la forma, el color, etc. Los algoritmos que emplean más datos además del nivel de gris se conocen como algoritmos de contexto.

La clasificación contextual puede clasificarse en tres tipos en función de la etapa en la que hacen uso de las características espaciales: pre procesado, post procesado y clasificador contextual (Gong and Xu, 2004):

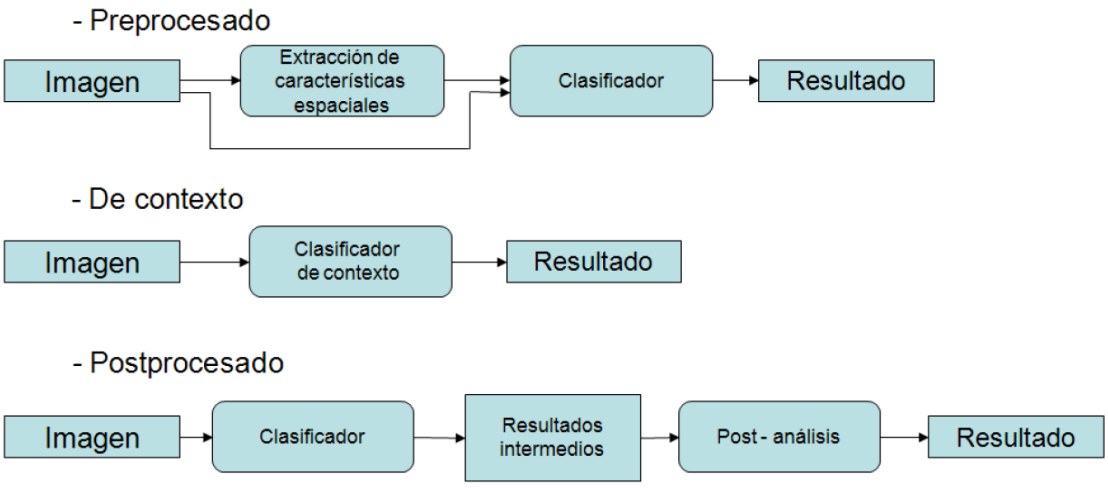


Fig. IX. Clasificadores de contexto. Adoptada de Gong y Xu,2004 (Regalado, 2018)**.**

- Métodos de pre procesado: Consisten en extraer de la imagen una serie de características espaciales descriptivas del dominio circundante de cada pixel.

  Cada una de las características espaciales extraídas de la imagen se incorporan a un espacio multidimensional de características, de modo que cada pixel tiene asociado un vector de características espectrales y espaciales o texturales. Posteriormente, cada pixel será asignado a una clase por un clasificador de máxima probabilidad, de mínima distancia o cualquier otro clasificador estadístico o no paramétrico.

Existen numerosos métodos diseñados para extraer información textual de la imagen, como, por ejemplo, la matriz de concurrencia de niveles de gris (Haralick, Shanmugan and Dinstein, 1973), los filtros de Gabor (Weldon and Higgins,1998), la transformada wavelet (Ruiz, Fernández and Recio, 2014), el vario grama (Chica and Abarca,2000).

- Métodos de post procesado: Estos métodos perfeccionan las imágenes ya clasificadas y son conocidos como re clasificadores contextuales. Los métodos de post procesado evalúan una clasificación y la modifican según un determinado criterio. Un ejemplo de algoritmo de post procesado es la aplicación de un filtro de moda, que asigna a cada pixel la clase más frecuente en su vecindario. Otro método de post procesado es el método SPARK o *kernel* de reclasificación espacial, que consiste en definir unas matrices que representan patrones espaciales típicos de los distintos usos de suelo (Sluiter *et al,*2004).
- Clasificadores de contexto: En lugar de extraer información contextual y almacenarla para usarla en la clasificación como ocurre en los métodos contextuales de pre procesado, un clasificador contextual organiza la información del entorno del pixel directamente en el proceso de asignar una etiqueta de clase al pixel.

Clasificadores orientados a objetos

Entre los métodos clasificadores que consideran la distribución espacial existente en la imagen se incluyen los clasificadores orientados a objetos.

Hay y Castilla (2006), definen el Análisis de Imágenes Orientado a Objetos como una disciplina dedicada a dividir las imágenes en objetos con significado propio y al mismo tiempo, obtener sus características desde un punto de vita espacial, espectral y temporal. En esta metodología se considera la forma, la textura y las propiedades espectrales de los objetos que forman la imagen, así como las relaciones existentes con los objetos vecinos situados en un contexto espacial más o menos cercano, aumentando de forma considerable las características descriptivas de los objetos que facilitaran su correcta clasificación, siendo los objetos los que se clasifican y no los pixeles individualmente.

El primer paso de estos clasificadores es la segmentación de la imagen que debe hacerse teniendo en cuenta la resolución de la imagen y el tamaño de los objetos a identificar. El

resultado es un conjunto de regiones que cubren totalmente la imagen. Todos los pixeles de una región son similares con respecto a alguna característica, al mismo tiempo que son diferentes de los pixeles situados en regiones adyacentes. Una imagen puede segmentarse en objetos de mayor o menos tamaño, determinando las características derivadas de los objetos de la imagen. El hecho de segmentar una imagen en diferentes escalas da lugar a que surja una estructura jerárquica entre los objetos de los distintos niveles, ya que un objeto puede incluir objetos de niveles inferiores, y a su vez, formar parte de objetos de un nivel superior, por ejemplo, se puede segmentar una imagen y obtener objetos de tipos de árboles, que a su vez pudieran segmentarse en objetos de tipo hija, si la resolución espacial lo permite; y a su vez los objetos arboles forman parte de un objeto tipo bosque.

La característica más valiosa de la clasificación de imágenes orientada a objetos es la posibilidad de obtener de un gran número de características descriptivas de los objetos y de las relaciones existentes entre los mismos que permiten describirlos mejor, y, por lo tanto, diferenciarlos y obtener resultados más precisos y específicos (Recio, 2010).

4.4.9.2 Métodos de clasificación no paramétricos

Los métodos de clasificación pueden ser paramétricos o no paramétricos. Los clasificadores paramétricos asumen que los datos de una clase siguen una distribución normal y que los parámetros estadísticos, como la media y la varianza, difieren significativamente entre las clases.

Sin embargo, esta sensación no siempre es correcta, ya que generalmente existe confusión espectral entre clases parecidas (Romero and Calonge, 2004), y clases diferentes con propiedades espectrales similares (Vega, 2011).

Otro de los inconvenientes de los clasificadores paramétricos reside en la dificultad de combinar datos espectrales con otros datos auxiliares que permitan completar la información proveniente de la imagen, por no cumplir los requisitos estadísticos impuestos por estos métodos.

En cambio, los métodos no paramétricos no realizan ninguna asunción sobre la naturaleza de los datos ya que no emplean parámetros estadísticos para calcular la separabilidad entre clases. Además de ser especialmente adecuados para la incorporación de datos extremos a

las imágenes en el proceso de clasificación. Entre los clasificadores no paramétricos más utilizados están los sistemas expertos basados en arboles de decisión (Huang and Jensen, 1997) y las redes neuronales (Romero and Calonge, 2004; Vega,2011; García, 2013; Hernández, 2014; Mihaich, 2014).

4.5 Pre Procesamiento de Imágenes Digitales

4.5.1 Ruido en Imágenes

Las imágenes al ser adquiridas usualmente vienen contaminadas con algún tipo de ruido. El ruido puede deberse a varias causas, por ejemplo, durante el proceso de adquisición. Al tomar varias imágenes de una misma escena a través del mismo captor, aun en situación controladas, uno puede darse cuenta de que el valor de un pixel a lo largo de la secuencia de imágenes no es el mismo.

El ruido puede deberse a las siguientes causas:

1. Fluctuaciones espurias en los niveles de gris de los pixeles, introducción por el sistema de adquisición.
2. Fluctuaciones aleatorias o inexactitudes en los datos, limitada por el equipo de cómputo usado o redondeo.
3. Agrupamientos erróneos de partes de los objetos en las imágenes.

4.5.1.1 Tipos de Ruido

Como se ha mencionado, en muchas ocasiones dos o más imágenes pueden venir contaminadas con ruido al ser adquiridas por un sensor. Esto puede afectar la operación del sistema de detección e identificación de objetos. Algunos tipos de ruido que suelen usarse para probar el desempeño de filtros son:

Sea *f (x, y)* una imagen adquirida mediante una de las cámaras del robot, diremos que esta imagen puede venir alterada o contaminada con ruido mezclado. Una versión ruidosa $\tilde{f}(x, y)$ de la imagen de entrada *f (x, y)* se obtiene al sumar o restar al azar a cada pixel de *f (x, y)* un entero *c*, tal que $\tilde{f}(x, y) = f(x, y) \pm c$.

Una manera de generar una versión ruidosa $\tilde{f}(x,y)$ de una imagen *f (x, y)*, es:

$$\tilde{f}(x,y) = \begin{cases} f(x,y) & a < l* \\ f(x,y) + bc & a \geq l* \end{cases}$$

En este caso, $a \in [0,1]$ es una variable aleatoria uniformemente distribuida, el parámetro *l** controla cuanto la señal *f (x, y)* es contaminada, la variable *c* determina que tan severo es el ruido. Cuando *b=1* el ruido adicionado a *f (x, y)* es de tipo aditivo. De igual forma, cuando *b=-1* el ruido adherido al patrón es de tipo substractivo.

En el caso de imágenes con valores en el rango [0, *l-1*], el valor de *c*, sustraído o adicionado a cada pixel será tal que:

$$0 \leq f(x,y) \pm c \leq l - 1$$

Cuando no se conoce la naturaleza del ruido presente en una imagen, se asume que este sigue un comportamiento de tipo gaussiano. En este caso la variable *c* toma valores de acuerdo a la siguiente ecuación:

$$c = e^{-\frac{z^2}{2\sigma^2}}$$

Donde $z \in \mathbb{R}_+ \cup 0, y\, \sigma \in \mathbb{R}_+$ es un parámetro de amplitud gaussiana.

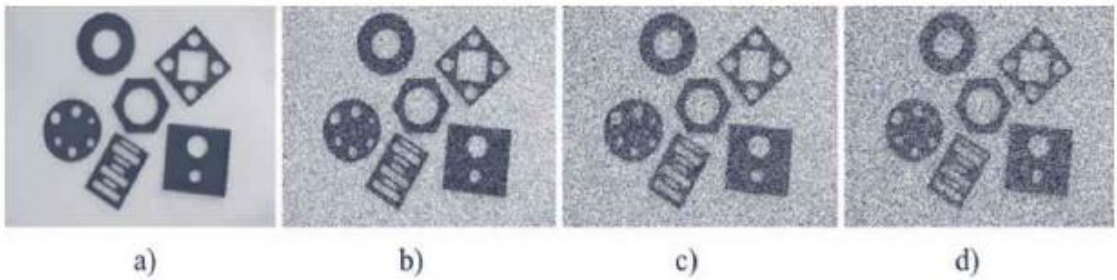


Fig. X. Imagen con seis objetos a) imagen original b), c) yd) son tres versiones de la imagen, pero contaminadas con ruido gaussiano, $\sigma = 0.1$, $\sigma = 0.2, \sigma = 0.3$, respectivamente (Humberto S. Azuela, 2020)**.**

En la figura anterior, se muestra una imagen y tres versiones ruidosas con ruido mezclado gaussiano para tres valores de σ. Conforme el valor de σ aumenta, el efecto del ruido sobre la imagen incrementa a una perdida cada vez más notoria de la información.

Otro tipo de ruido que suele acompañar a las imágenes es el tipo *Sal y Pimienta*. Este ruido se caracteriza por saturar el valor de uno o más pixeles de *f (x. y)* hacia el extremo positivo (ruido tipo sal, *l-1)* o hacia el extremo inferior (ruido pimienta, 0). Para generar ruido saturado tipo *Sal y Pimienta* usando la ecuación:

$$\tilde{f}(x,y) = \begin{cases} f(x,y) & a < l* \\ f(x,y) + bc & a \geq l* \end{cases}$$

Es suficiente que para cada pixel con coordenadas *(x, y)*, *b=1* y *c=0* (pimienta) o *c=l-1 (*sal*),* con *l* el valor máximo de saturación que cada pixel puede tomar. A continuación, se muestran tres versiones ruidosas con ruido mezclado Sal y Pimienta de la imagen mostrada con anterioridad, para tres valores del parámetro *l**. Al igual que en el caso del ruido gaussiano, nótese que conforme el valor del parámetro *l** aumenta, más notorio es el efecto del ruido sobre la imagen.

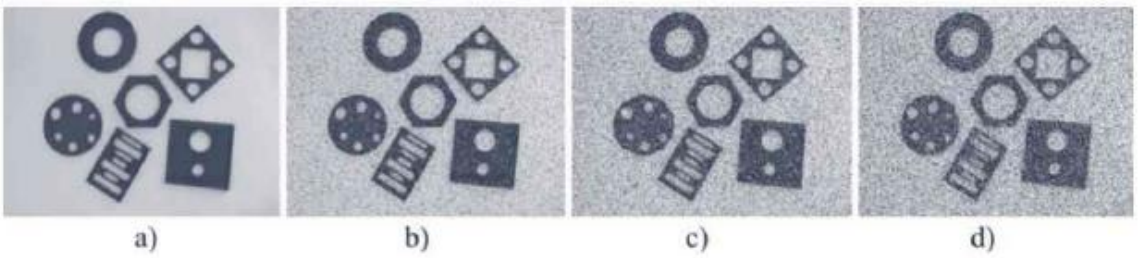


Fig.XI. Imagen con seis objetos (a) Imagen Original y (b-c-d) son tres versiones contaminadas de dicha imagen con ruido sal y pimienta, las cuales corresponden a los casos *l*=0.1, l*=0.2* y *l*=0.3,* respectivamente (Humberto S. Azuela, 2020).

Para poder resolver este tipo de problemas con las imágenes de muestra, existen las técnicas de filtrado, el objetivo de estas consiste en reducir o eliminar, en el mejor de los casos, el

ruido presente en una imagen. Debido a que el proceso de modelado del ruido es algo muy complicado, en un escenario ideal se obtiene una reducción del mismo presente en la señal. Existen distintos métodos de filtrados para reducir la influencia del ruido en varias ocasiones presentes en las imágenes adquiridas a través de los sensores utilizados. Para llevar a cabo esta investigación, se centró en dos métodos principalmente el *Filtro Gaussiano* y el *Filtrado de Difusión Anisotrópica*, los cuales se presentan a continuación.

4.5.2 Filtro Gaussiano

Simulan una distribución gaussiana bivariante. El valor máximo aparece en el pixel central y disminuye hacia los extremos tanto más rápido cuanto menor sea el parámetro de desviación típica *s*. El resultado será un conjunto de valores entre 0 y 1. Para transformar la matriz a una matriz de números enteros se divide toda la matriz por el menor de los valores obtenidos. La ecuación para calcularla es:

$$g(x,y) = e^{\frac{x^2+y^2}{2*s^2}}$$

$$G(x,y) = \frac{g(x,y)}{min_{x,y}(g(x,y))}$$

1	4	7	4	1
4	20	33	20	4
7	33	55	33	7
4	20	33	20	4
1	4	7	4	1

Tabla III. Filtro Gaussiano con *s=1* y *r=2*

0	0	0
0	1	0
0	0	0

DIV1

Tabla IV. Matriz de filtrado identidad.

Son una manera de obtener filtros de tipo genérico. Pueden ser útiles, por ejemplo, cuando se asume que la respuesta de un pixel es función de la reflectividad de los pixeles vecinos atenuada en función de la distancia. El alcance de esta atenuación (*r*) viene marcado por el tamaño de la ventana de filtrado (*w= 2r + 1)* que debe especificarse previamente.

4.5.2 .1 Filtro Promedio Gaussiano

Es un tipo de filtrado diferente a la técnica Filtrado Promedio Aritmético h_{pa}, es denominado también, filtro promedio gaussiano representado por h_{pg}, el cual incluye una forma estructural exponencial de las coordenadas cuadráticas (*x, y)* de la imagen. El filtro promedio gaussiano se define como:

En el dominio de los pixeles, un filtro promedio gaussiano h_{pg}, *es un arreglo de* $h_x\, h_x$ *elementos o peros w1, w2…, w (h x h), tales que* $w_i > 0$, *son ajustados en términos de:*

$$h_{pg} = H(x,y) = c\, e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Donde c es una constante de normalización.

Por lo que, de la ecuación anterior, podemos obtener lo siguiente:

$$\frac{H(x,y)}{c} = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Una vez seleccionado el valor de σ^2 se puede evaluar el núcleo para la ventana *h*.

Dada una imagen *f (x, y)* de dimensiones *n x m* pixeles, un procedimiento sencillo para filtrar a la imagen *f (x, y)* usando un filtro promedio gaussiano establecido por la ecuación anterior y utilizando una máscara de dimensión *h x h* pixeles para obtener la correspondiente imagen filtrada *g* (*x, y),* es el que se a continuación se describe:

Pseudocódigo. Algoritmo de filtrado promedio gaussiano de imágenes.

Procedimiento para filtrar una imagen $f(x,y) \in \mathbb{R}^{n \times m}$ con máscara $h_{pg}(i,j) \in \mathbb{R}^{h \times h}$.

Entrada: imagen en niveles de gris $f(x,y) \in \mathbb{R}^{n \times m}$.

Salida: imagen filtrada $g(x,y) \in \mathbb{R}^{n \times m}$.

```
g(x,y)=zeros(n, m); % la imagen g(x,y) se llena de ceros.
Diseñar máscara h_pg(i,j) de tamaño h × h, con valores positivos.
for y=h hasta y=(m-1)
    for  x=h hasta x=(n-1)
        v=0;
        for i=1 hasta i=h
            for  j=1 hasta j=h
                v=v+(f(x+i+1-h, y+j+1-h)*h_pg(i,j));
            end
        end
        g(x,y)=trunc(v);
    end
end
% fin del procedimiento.
```

Para obtener el filtro gaussiano con el parámetro $\sigma^2 = 2$ y dimensión de la ventana 7 x 7 se procede: Considere una ventana de dimensión 7x7 con los siguientes datos *x= {-3,-2,-1,0,1,2,3},* así como *y= {-3,-2,-1,0,1,2,3}*, entonces los valores de la ventana para el filtro establecido por: $\frac{H(x,y)}{c} = e^{-\frac{x^2+y^2}{2\sigma^2}}$ son los que se muestran en la Tabla III.

La máscara de la tabla III utilizada de forma directa sobre la imagen *f (x, y)* de la figura “” reduce la influencia del ruido sobre la imagen, obteniendo la imagen filtrada de salida *g (x, y)* tal y como se muestra a continuación.

Fig.XII. Imagen Original a). En la figura b) se muestra la imagen filtrada de salida *g (x, y)* procesada a través de un filtro promedio gaussiano, usando $\sigma^2 = 2$ y una ventana de 7x7 (Humberto S. Azuela, 2020)**.**

(x, y)	-3	-2	-1	0	1	2	3
-3	0.0111	0.0388	0.0821	0.1054	0.0821	0.0388	0.0111
-2	0.0388	0.1353	0.2865	0.3679	0.2865	0.1353	0.0388
-1	0.0821	0.2865	0.606	0.7788	0.606	0.2865	0.0821
0	0.1054	0.3679	0.7788	1.0	0.7788	0.3679	0.1054
1	0.0821	0.2865	0.6065	0.7788	0.6065	0.2865	0.0821
2	0.3688	0.1353	0.2865	0.3679	0.2865	0.1353	0.0388
3	0.0111	0.0388	0.0821	0.1054	0.0821	0.0388	0.0111

Tabla V. Filtro Gaussiano, con $\sigma^2 = 2$ **y *h*=*7*** (Humberto S. Azuela, 2020)**.**

4.5.3 Filtrado de Difusión Anisotrópica

Se habla de técnicas de filtrado basadas en la difusión cuando las operaciones de limpieza de ruido, suavizado y realce de bordes se modelan como un proceso de difusión entre celdas adyacentes, de manera que con el paso del tiempo las zonas similares se vuelven más homogéneas y los bordes se realzan. Si la difusión de la que se habla es isótropa la difusión

se produce por igual en todas las direcciones. Si, por el contrario, se habla de anisotrópica entonces la difusión variara según la dirección.

4.5.3.1 Evolución

Se han definido diversos modelos para el suavizado y mejora de bordes de las imágenes basados tanto en procesos de difusión lineal como no lineal. Estos intentos surgen de la idea de analizar las imágenes a diferentes niveles de resolución. El primer modelo fue propuesto por Witkin y posteriormente mejorado por Koenderink. En este modelo la imagen resultante se obtenía a partir de la convolución de la imagen original con un *kernel gaussiano* de una cierta varianza σ_t:

$$I(x, y, t) = I_0(x, y) * G(x, y; \sigma_t)$$

Cambiando la varianza σ_t, se obtiene un conjunto de imágenes a distintas resoluciones Fig. XIII.

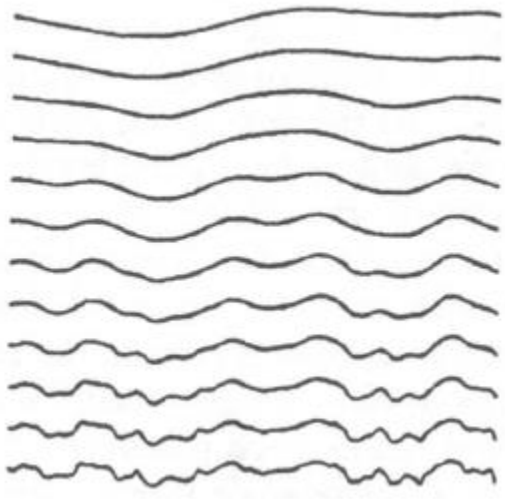


Fig. XIII. Representación de las señales obtenidas mediante la convolución de la original (señal de más abajo) con un *kernel gaussiano*, cuya varianza se va incrementando (desde abajo hacia arriba) (Filtrado de Difusión Anisótropica, 2007)**.**

El problema de esta técnica propuesta, es que las imágenes resultantes tenían normalmente unas resoluciones muy bastas, que impedían determinar exactamente la localización de las

fronteras y bordes de las estructuras. Además, los bordes que se veían en estas resoluciones estaban cambiados con respecto a sus posiciones originales.
Koenderink apuntó que esta serie de imágenes obtenidas derivadas de un solo parámetro se podían ver como las soluciones de la ecuación de difusión del calor:

$$I_t = \nabla I = (I_{xx} + I_{yy})$$

Con la condición inicial $I(x, y, 0) = I_0(x, y)$, la imagen original. Koenderink formulo la ecuación de difusión debido al planteamiento de dos criterios:

1. Causalidad: Al disminuir la resolución no pueden generarse espúreos en la imagen.
2. Homogeneidad e isotropía: La borrosidad debe ser invariante en el espacio.

Estos dos criterios conllevan a la formulación de la ecuación del calor, pero el segundo criterio se puede sustituir por algo más útil. Esto fue lo que hicieron P. Perona y J. Malik, consiguiendo así desarrollar un modelo de suavizado multi escala y mejora de bordes, que ha supuesta una herramienta muy útil para la limpieza de ruido en la imagen. Perona y Malik enunciaron los criterios que se debían cumplir para obtener un buen modelo de descripción de imágenes multi escala:

1. Causalidad: De la misma manera que estableció Koenderink.
2. Localización inmediata: En cada resolución los límites de las regiones deben ser distinguidos y deben coincidir con los originales.
3. Suavizado según zona: El suavizado debe ser mayor dentro de una región que con las regiones vecinas.

4.5.3.2 Base Matemática

Teniendo en cuenta esto, se definió un nuevo modelo de filtrado de difusión que lógicamente tiene como base matemática la ecuación de difusión del calor. Esta ecuación expresada para el procesado de imágenes se representaría del siguiente modo:

$$\frac{\partial I(x, y, t)}{\partial t} = div(c(x, y, t)\nabla I(x, y, t))$$

Donde *I (x, y)* representaría la función de la intensidad de la imagen, *I (x, y, t)* es la evolución de la imagen en el tiempo y *div* es el *operador divergencia*, definido:

$$divf(\vec{x}) = \sum_{i=0}^{n-1} \frac{\partial f}{\partial x_i}$$

Y el operador ∇ denota el *gradiente:*

$$\nabla f(\vec{x}) = (\frac{\partial f}{\partial x_0}, \dots, \frac{\partial f}{\partial x_{n-1}})$$

La función *c (x, y, t)* se conoce con el nombre de *coeficientes de difusión* y junto con el *gradiente* describe el flujo:

$$\Phi = c(x, y, t)\nabla I(x, y, t)$$

Koenderink estableció que el coeficiente *c* debía ser una constante independiente de la localización espacial. Es este el caso en el que se realiza un filtrado lineal, dando lugar a un flujo isótropo y es cuando además de eliminar ruido se eliminan detalles. Perona y Malik establecieron que el coeficiente *c* debía variar según su localización espacial. De esta manera se obtiene entonces un flujo anisótropo correspondiente a un filtrado no lineal.

4.5.3.3 Propiedades de la Difusión Anisótropa

El primer criterio que debe cumplir la difusión anisótropa es el criterio de causalidad. Este criterio establece que no se deben introducir nuevos detalles en la imagen al pasar de una resolución fina a una menos fina. También se puede definir diciendo que todos los máximos y todos los mínimos de la función intensidad de la imagen existentes en las distintas resoluciones deben pertenecer a la imagen original.

La ecuación de difusión es un caso particular de ecuación elíptica. Estas ecuaciones satisfacen el *principio del máximo*, que establece que todos los máximos de la solución de la ecuación en el espacio y en el tiempo pertenecen a la condición inicial (imagen original). *I*

(x, y, t) obedece este principio debido a que el *coeficiente de difusión* nunca es negativo y a que es diferenciable. Además, si las condiciones de frontera son adiabáticas los máximos solo pertenecen a la condición inicial. También se puede demostrar sencillamente que se cumple el *principio del mínimo*. Por tanto, se cumple el criterio de causalidad.

Con las técnicas de filtrado convencionales como el filtrado paso bajo y el filtrado basado nen la difusión lineal el precio que se paga por le eliminación de ruido y mejora de resolución es la difuminación de los bordes. Esto hace que su detección y localización sea más difícil. Para la reconstrucción y mejora de los bordes de la imagen difuminada se puede hacer un filtrado paso alto o filtrar mediante el proceso de difusión hacia atrás en el tiempo. Esto tiene el problema de que en la mayoría de los casos se llega a un estado inestable.

En el caso de la difusión anisótropa si se escoge de forma apropiada el *coeficiente de difusión* se puede que la difusión anisótropa mejore los bordes de la imagen, filtrando hacia adelante en el tiempo, con la estabilidad que garantiza el *principio del máximo*.

Si se modela un borde como una función escalón convolucionada con una gaussiana y se asume que el borde esta alineado con el eje de coordenadas *y*, la expresión del *operador divergencia* se puede simplificar a:

$$div(c(x,y,t)\nabla I) = \frac{\partial(c(x,y,t)I_x)}{\partial x}$$

Si según lo aconsejado se escoge *c* como una función del *gradiente* de la intensidad de la imagen *I:*

$$c(x,y,t) = g(I_x(x,y,t))$$

Y si denotamos como ya se ha visto antes el flujo $\Phi = cI_x$, entonces la ecuación de difusión para 1D se escribiría del siguiente modo:

$$I_t = \frac{\partial\Phi(I_x)}{\partial x} = \Phi'(I_x)I_{xx}$$

Lo que interesa es la variación en el tiempo de la pendiente del borde: $\frac{\partial I_x}{\partial t}$. Si el coeficiente de difusión es mayor que cero (*c>0)* la función *I ()* se suaviza, y el orden de la diferenciación debe ser invertido:

$$\frac{\partial I_x}{\partial t} = \frac{\partial I_t}{\partial t} = \frac{\partial}{\partial t}\left(\frac{\partial \Phi(I_x)}{\partial x}\right) = \Phi'' I_{xx}^2 + \Phi' I_{xxx}$$

4.6 Mecanismos de la Visión Humana

4.6.1 El Ojo

El ojo es el detector de las señales visuales. Efectúa la focalización de las imágenes provenientes del exterior para que la retina pueda recibir la imagen. Esta realiza un pretratamiento de la imagen y envía la información a la corteza visual a través del nervio óptico y la vía visual en el cerebro. En la Fig. XIV se muestran los mecanismos básicos de la formación de una imagen sobre la retina. La luz pasa a través de la córnea y el humo acuoso antes de penetrar, atravesando la pupila, dentro del cristalino y el humor vitreo. La imagen sufre una refracción debido a todos estos componentes del ojo, antes de que estimule a la retina. Los defectos existentes en el cristalino (existentes aun en personas que tienen una visión normal) producen una aberración esférica que produce imágenes borrosas. Este comportamiento puede considerarse como un filtro paso bajo bidimensional.

El diámetro de la pupila varía entre 2 y 9 mm, dependiendo de la iluminación. Esta variación puede multiplicar la cantidad de luz que llega a la retina por un factor de 16 para compensar las iluminaciones débiles, pero esta compensación se hace a expensas de la calidad de la imagen retiniana. Esta apertura se puede ver igualmente como un filtro paso bajo. La frecuencia de corte más alta aparece cuando el diámetro de la pupila es de 2 mm.
Un agrandamiento progresivo del diámetro produce una frecuencia de corte cada vez más baja. Si la iluminación es suficientemente alta, los foto-receptores de la retina aumentaran su sensibilidad gracias a una adaptación química y el diámetro de la pupila disminuirá significativamente. Este mecanismo trata de compensar las aberraciones esféricas y cromáticas del ojo y permite que el ojo responda a iluminaciones que comprenden 9 niveles de magnitud.

4.6.2 La retina

La retina es la capa sensorial del ojo. Ella lleva a cabo la transducción de una señal electromagnética (la imagen) a electroquímica (los impulsos nerviosos) y efectúa un pretratamiento de la información antes de enviarla al cerebro por medio del nervio óptico.
El tejido retiniano está compuesto de 5 tipos de células que están organizadas en capas. La capa más alejada de la fuente de luz es la capa transductora. Esta estructura está compuesta de 2 tipos de células: los conos y los bastones. Existen alrededor de 130 millones de bastones y 6.5 millones de conos en la retina.

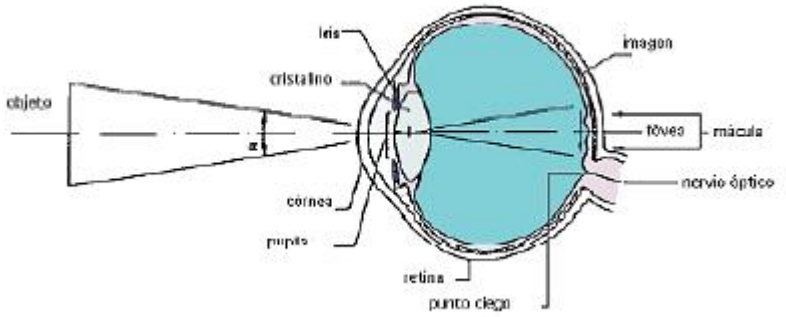


Fig.XIV. Vista esquemática del ojo (Vilet, 2005)**.**

Los bastones reaccionan a las formas y las estructuras en movimiento y pueden funcionar con niveles bajos de luminancia (visión escotópica). Los conos necesitan niveles elevados de luminancia (visión fotópica) y detectan el color y los detalles. Están distribuidos alrededor del eje ocular principalmente.

4.6.3 Codificación Neuronal

Aun cuando la codificación del color se efectué al nivel de los receptores, una gran parte de la codificación global de la imagen se lleva a cabo en la retina directamente, gracias a las conexiones mencionadas anteriormente. El propósito de esta red es el de permitir que el sistema visual se adapte a la iluminación de la escena de tal manera que los objetos de esta escena sean transmitidos al cerebro con fidelidad, aun bajo condiciones de iluminación cambiante. Ya que el nervio óptico tiene una capacidad limitada para transferir la información, la retina puede verse como un sistema que efectúa una comprensión del rango dinámico de la imagen. Esta característica es utilizada por algoritmos que utilizan un modelo de la visión

humana para la comprensión de imágenes. El resultado de los estudios de la adaptación a los niveles de iluminación y de los fenómenos como el de inhibición lateral ha sido el de proponer modelos con dos canales de organización de la información espacial en la retina. El primer canal contiene la información sobre las características promedio de la iluminación de la imagen y puede verse como el resultado de un filtrado paso bajo. Este canal proporciona a su vez información a las células receptoras que responden a la iluminación local. Esto constituye un método rápido de adaptación de su sensibilidad y que permite un aumento de su rango dinámico de 1 a 3 órdenes de magnitud. El segundo canal está formado de las respuestas de las células receptoras, eliminando las bajas frecuencias. Se comporta como un filtro paso banda. Mientras que el canal paso bajos proporciona información sobre el nivel de contraste en la imagen, el canal paso banda transmite información sobre los bordes y las líneas de la imagen. Esta última información es importante para distinguir los objetos en el mundo exterior y se envía a los centros visuales en el cerebro.

4.6.4 Células Ganglionares y Cuerpo Geniculado Lateral

Las células ganglionares y las otras células en las estructuras superiores de la vía visual responden a estímulos específicos que caen sobre la retina. El estímulo más eficaz sobre las células ganglionares y sobre las células del cuerpo geniculado lateral es un disco luminoso de aproximadamente 2 mm de diámetro, debido a las conexiones en paralelo y de retroalimentación. Este estimulo debe localizarse en un lugar particular del campo visual denominado campo receptor. Este campo en una célula ganglionar es la zona de la retina que envía información a la célula en particular. Los campos receptores son circulares y están constituidos ya sea de un centro excitatorio y una periferia inhibitoria (denominados zona “on”) o de una configuración inversa (zona “off”). Un disco luminoso que cubra exactamente el centro de una célula excitatoria “on” constituye el estímulo más eficaz.

Una barra luminosa constituye un estímulo eficaz si ella cubre una parte importante de la región excitatoria “on” y solo una pequeña región de la periferia inhibidora. Las células ganglionares no toman en cuenta el nivel absoluto de iluminación, ya que miden diferencias entre el nivel de iluminación de una pequeña región y la iluminación media de su entorno inmediato. La información proporcionada por la retina se transmite al cerebro promedio de alrededor del millón de fibras nerviosas (axones de las células ganglionares) y llega, después

de dividirse en el quiasmo óptico, a los cuerpos geniculados laterales que reaccionan de una manera muy similar a las células ganglionares. La independencia de los estímulos hacia la orientación se mantiene hasta este nivel de organización en la vía visual.

4.6.5 La Corteza Visual

La corteza visual primaria ocupa la mayor parte posible de los lóbulos occipitales del cerebro. Está constituida de alrededor de 1010 células organizadas en capas construidas solamente a partir de algunos tipos de neuronas, denominadas neuronas simples, complejas, hipercomplejas e hipercomplejas de orden superior. Las neuronas corticales “simples” responden a barras luminosas sobre la retina en lugar de responder a discos luminosos. Estas células se comportan como si su entrada proviniese de muchas células con respuesta del tipo centro-periferia. El campo receptor de este tipo de neurona es elíptico con una respuesta máxima para una orientación especifica de luz proveniente de una región particular del campo visual. Si la barra luminosa sufre una rotación, otras células comenzaran a responder de acuerdo al ángulo de orientación.

A este nivel del sistema visual comienza el procesamiento especifico de la información, dependiendo de la orientación de los estímulos. Las células complejas se comportan como si su información proviniera de múltiples células simples, todas con campos receptores de la misma orientación, pero viniendo de localizaciones ligeramente diferentes. Las respuestas de las células hipercomplejas son similares, pero además de la orientación especifica, debe haber una discontinuidad en su campo receptor, tal como el fin de una línea o una esquina de una figura geométrica. La Figura “” muestra la respuesta de estas células en un rectángulo blanco sobre un fondo oscuro.

4.6.6 Modelos del Sistema Visual

Un análisis de la corteza muestra una organización en columnas para la dominancia ocular y para la respuesta a la orientación. Un pequeño islote de corteza de un 1 mm cuadrado de superficie y 2 mm de

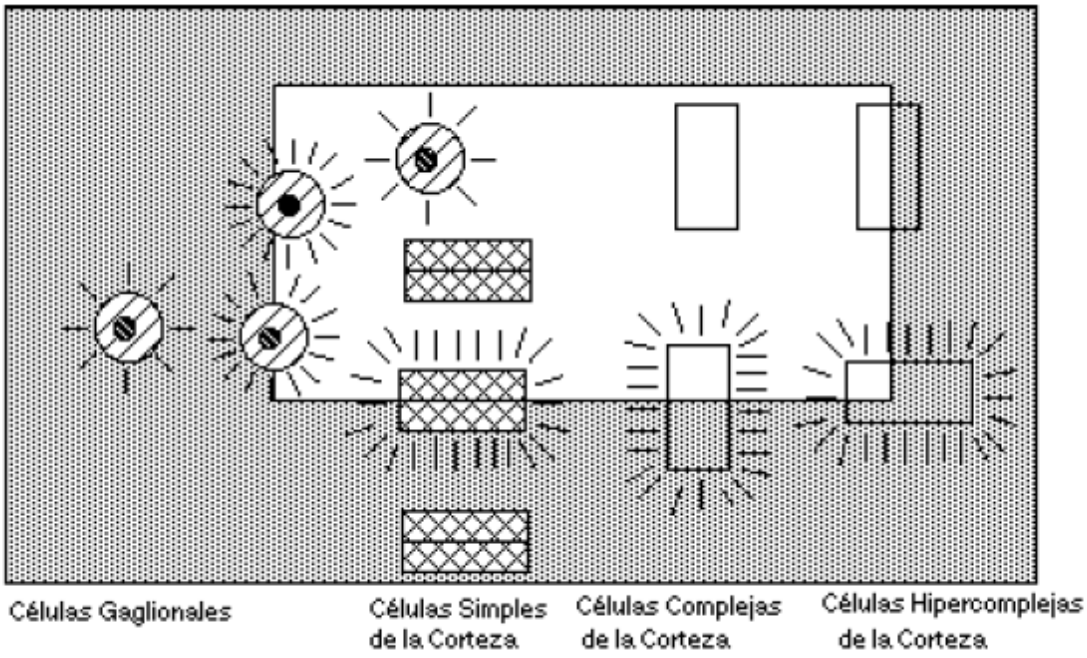


Fig XV. Respuesta de las células corticales a un rectángulo blanco (Vilet, 2005)**.**

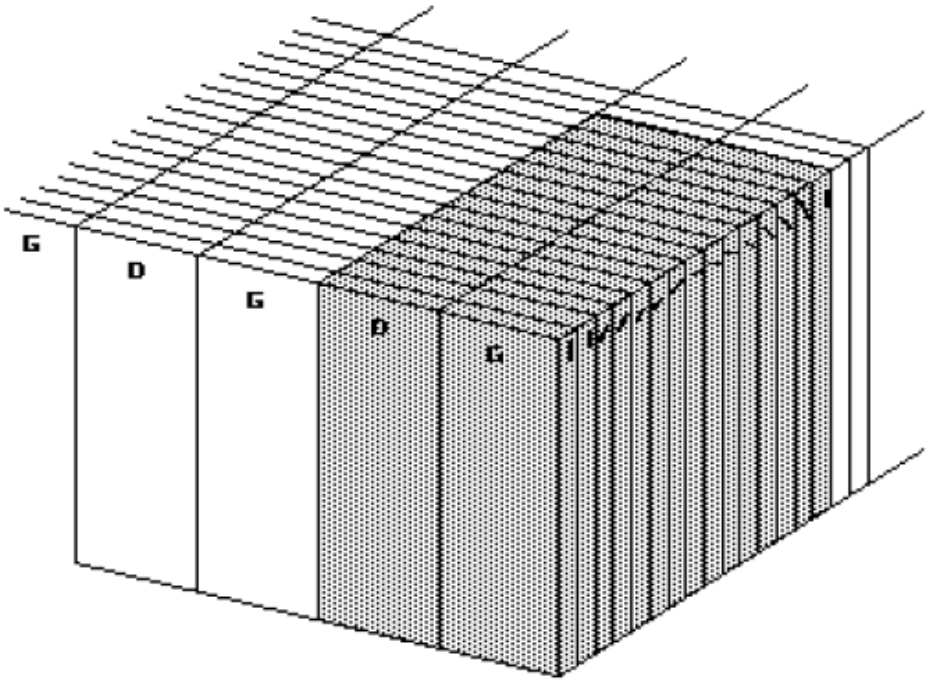


Fig.XVI. Modelo de columnas de Hubel y Wiesel (Vilet, 2005)**.**

Espesor puede considerarse como el elemento base de la corteza primaria. El islote contiene un conjunto de hojuelas que corresponden a todas las orientaciones posibles, cuantificadas en treinta direcciones y otro conjunto de hojuelas con dominancia ocular alteradamente izquierda y derecha. La figura XVII muestra este modelo de la corteza organizada en columnas [Hubel & Wiesel,1968].

El efecto de la pupila, la aberración esférica del cristalino y la limitación en frecuencia debida al número limitado de fotoreceptores puede modelarse como un filtro paso bajas. Posteriormente, ¿las características no lineales de los fotoreceptores pueden verse como una respuesta logarítmica o como si siguieran una curva del tipo L? al nivel de la retina, a estas transformaciones sigue un filtro paso alto, que corresponden al fenómeno de inhibición lateral de las células ganglionares (y también a las células de los cuerpos geniculados laterales). Finalmente, el procesamiento a nivel de la corteza visual puede verse como un conjunto de filtros direccionales. El modelo que reúne a estas etapas ha sido propuesto por kunt [1985]. En este sistema existe una cierta anisotropía. La sensibilidad esta disminuida en las direcciones oblicuas con respecto a las direcciones vertical y horizontal. Para dos señales sinusoidales, la sensibilidad disminuye 3dB a 45 & deg; de rotación con respecto a la línea horizontal.

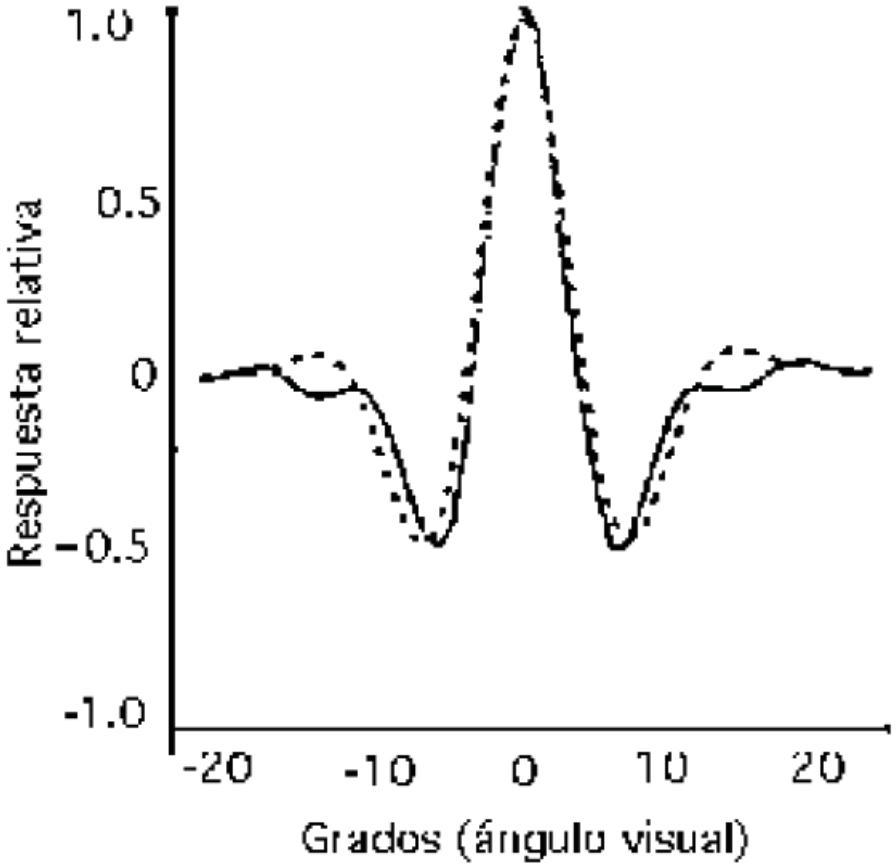


Fig. XVII. Función de transferencia de una célula simple y su síntesis por una función de Gabor (según Mallat) (Vilet, 2005)**.**

Esta propiedad se utiliza en ciertos tipos de codificación de imágenes, donde las componentes diagonales se cuantifican con menos resolución que las componentes de las otras direcciones. Aun cuando el modelo multidireccional del sistema visual se aproxima bien a los datos experimentales, especialmente al nivel de las células simples en la corteza visual, se desconoce el tipo de información que pueden extraerse de esta descomposición y su relación con los procesamientos efectuados a nivel de células complejas e hipercomplejas. Sin embargo, se pueden mostrar que las respuestas al impulso de las células simples pueden modelarse por gaussianas moduladas por señales sinusoidales que generan un tipo particular de transformada de Fourier con ventana deslizante, denominada transformada de Gabor.

Para una posición *u* y para una frecuencia ω, la transformada de Fourier dentro de una ventana *g* de una función *f(x)* se define como:

$$Gf(w,u) = \int_{-\infty}^{\infty} e^{-jwx} g(x-u) f(x) dx$$

Esta función mide, de una manera local alrededor del punto *u*, la amplitud de la componente sinusoidal de frecuencia ω y puede considerarse como la respuesta al impulso de un filtro paso bajos. Para la transformada de Gabor, la función *g(x)* es gaussiana. La figura anterior muestra la función de transferencia de una célula simple y en punteado, la función de Gabor que la modeliza. Esta modelización sirve como base a varios métodos de comprensión de imágenes, principalmente los métodos de descomposición multifrecuencial y las transformadas en "wavelets".

4.7 Inteligencia Artificial (IA)

¿Qué es la Inteligencia Artificial?

Antes de definir el termino Inteligencia Artificial, se debe primero reflexiona acerca de lo que es la inteligencia natural o simplemente inteligencia. Inteligencia es un término vago. Su definición ha dependido del tipo de individuo involucrado, del momento, incluso de las circunstancias. Platón, por ejemplo, decía que la inteligencia es la actividad que permite adquirir ciencia, lo cual, por supuesto es muy restrictivo. Él propuso dos tipos de inteligencia, pero de acuerdo con Howard Gardner hay nueve, según la psicología hay doce. En resumen, se puede decir que la inteligencia es una capacidad mental muy general que implica:

- Habilidad para razonar
- Planificar
- Resolver Problemas
- Pensar de forma abstracta
- Comprender ideas complejas
- Aprender con rapidez
- Aprender de la experiencia

La IA no supone un simple aprendizaje de un texto, cierta habilidad académica especifica, o resolver un test o examen de forma hábil. Mas bien, refleja una capacidad amplia y profunda para:

- La comprensión del entorno
- Ser capaz de capturar el significado de las cosas y darles un sentido, o para ingeniárselas a la hora de saber que hacer.

Definición. *Tomando esto como base, la Inteligencia Artificial puede definirse como la ciencia e ingeniería de las maquinas que actúan de manera inteligente.*

En este sentido, una maquina es inteligente cuando es capaz de tomar decisiones apropiadas en circunstancias inciertas. Otra manera de verlo es que una maquina se dice ser inteligente cuando es capaz de aprender a mejorar su comportamiento con base en sus experiencias.
Una vez dicho esto, no hay que confundir el computo basado en IA respecto de aquel fundamentado en el computo tradicional. Para entender esta diferencia, hay que considerar la pregunta: ¿Por qué hay ciertos problemas que son muy difíciles de resolver a través del cómputo tradicional?, recordar que el computo clásico, en sus inicios, fue diseñado para hacer dos cosas muy bien:

1. Realizar operaciones aritméticas de manera muy rápida.
2. Seguir de forma explícita un listado de instrucciones.

Tipos de Inteligencia Artificial

Se pueden diferenciar cuatro tipos de IA:

1) Inteligencia General (IG) o Inteligencia Fuerte: Llamada también Inteligencia Artificial General (IAG), incluso comparada con la inteligencia humana. Consiste en enseñarle a la maquina a aprender, a razonar y a planificar. No cubre todo el campo de la inteligencia, faltaría agregar a la maquina una mente, así como una conciencia, lo que implica estar consciente y un carácter propio, lo cual no es fácil de simular.

2) Super Inteligencia Artificial (SIA): UN paso más allá es la llamada Super Inteligencia Artificial puede alcanzarse relativamente rápido una vez que se llegue a la IAG, supuestamente porque los mayores obstáculos han sido sobrellevados.

3) Inteligencia Artificial Débil o aplicada (IAD): Se limita a afrontar tareas específicas, enfocadas a ayudar al ser humano. No intenta simular el rango completo de las habilidades cognitivas humanas.

4) Super Inteligencia Super Consciente IA (SI-SC-IA): Se refiere a la capacidad que tendrían criaturas existentes en algunos de los muy probables sistemas planetarios en nuestro Universo. Si partimos del hecho de que nuestra galaxia contiene alrededor de 400, 000 millones de estrellas, la probabilidad de que una de ellas haya dado origen a vida Super- Inteligente y Super. Consciente es muy alta.

4.7.1 Antecedentes de la IA

Se podría considerar que uno de los primeros pasos hacia la IA fue dado hace mucho tiempo por Aristóteles (384-322 A.C), cuando se dispuso a explicar y codificar ciertos estilos de razonamiento deductivo que el llamo *silogismos*. Otro intento sería el de Ramón Llull (d.C. 1235-1316), místico y poeta catalán, quien construyo un conjunto de ruedas llamado *Ars Magna*, el cual suponía iba a ser una maquina capaz de responder todas las preguntas.

Por su parte, Martin Gardner atribuye a Gottfried Leibniz (1646-1716) el sueño de "un algebra universal por el cual todos los conocimientos, incluyendo las verdades morales y metafísicas, pueden algún día ser interpuestos dentro de un sistema deductivo único". Sin embargo, no existió un progreso sustancial hasta que George Boole comenzó a desarrollar los fundamentos de la lógica proposicional. El objeto de Boole fue, entre otros: "Recoger… algunos indicios probables sobre la naturaleza y la constitución de la mente humana". Poco después, Gottlob Frege propuso un sistema de notación para el razonamiento mecánico y al hacerlo invento gran parte de lo que hoy se conoce como calculo proposicional (lógica matemática moderna).

En 1958, John McCarthy, responsable de introducir el término "Inteligencia Artificial", propuso utilizar el cálculo proposicional como un idioma para representar y utilizar el conocimiento en un sistema que denomino la "Advice Taker". A este sistema se le tenía que decir que hacer en vez de ser programado. Una aplicación modesta pero influyente de estas ideas fue realizada por Cordell Green en un su sistema llamado QA3.

Lógicos del siglo XX, entre ellos Karl Codel, Stephen Kleene, Emil Post, Alonzo Church y Alan Turing, formalizaron y aclararon mucho de lo que se podía y no podía hacerse con los

sistemas de lógica y de cálculo. En fechas más recientes, científicos de la computación como Stephen Cook y Richard Karp, descubrieron clases de cálculos que, aunque parecían posibles en principio, podrían requerir cantidades totalmente impracticables de tiempo y memoria (almacenamiento).

Gran parte del trabajo inicial se desarrolló en la década de 1960 y principios de los setenta en programas como General Problem Solver (GPS) de Allan Newell, Cliff Show y Herbert Simon. Otros sistemas que influyeron son: la integración simbólica, algebra Word, analogy puzles y control y robots móviles. Muchos de estos sistemas son el tema de un artículo llamado *Computer and Thought*.

Hacia finales de los setenta y principios de los ochenta, algunos programas que se desarrollaron contenían mayor capacidad y conocimientos necesarios para imitar el desempeño humano de expertos en varias tareas. El primer programa que se le atribuye la demostración de la importancia de grandes cantidades de conocimiento y dominio especifico es DENDRAL, un sistema de predicción de la estructura de las moléculas orgánicas que se considera su fórmula química y el análisis de espectrograma de masa. Le siguieron otros "sistemas expertos" como por ejemplo sistemas configurables y otros más.

En mayo 11 de 1997, un programa de IBM llamado *Deep Blue* derroto al actual campeón mundial de ajedrez, Garry Kasparov. Por otra parte, Larry Roberts desarrollo uno de los primeros programas de análisis de escena. Este trabajo fue seguido por una amplia labor de máquinas de visión (visión artificial). Otros proyectos que se pueden mencionar son CYC, una de cuyas metas era recolectar e interpretar una gran cantidad de información para su conocimiento. Aunque el interés en las redes neuronales se estancó un poco después de los trabajos pioneros de Frank Rosenblatt en los últimos años de la década de 1950, se reanudo con energía en los años ochenta. En la actualidad hay distintas aplicaciones con la IA.

Un breve resumen sobre la historia de la IA se puede describir a través de hechos relevantes como los que se describen a continuación:

384-322 AC: Las ideas más básicas se remontan a Aristóteles (384-322 a.C.). Fue el primero en postular un conjunto de reglas que describen una parte del funcionamiento de la mente para obtener conclusiones racionales.

250 AC: Ctesibio de Alejandría (250 a.C.) construye la primer maquina autocontrolada a través de un regulador de flujo de agua.

1315: Ramon Llull en su libro Ars Magna tiene la idea de que el razonamiento podía ser efectuado de manera artificial.

1936: Alan Turing diseña formalmente una maquina universal que demuestra la viabilidad de un dispositivo físico para implementar cualquier computo formalmente definido.

1943: Warren McCulloch y Walter Pitts presentan su modelo de neurona artificial.

1955: Herbert Simon, Allen Newell y J.C Shaw desarrollan el primer lenguaje de programación orientado a la resolución de problemas, el IPL-11. Un año más tarde desarrollan el LogicTheorist, el cual era capaz de demostrar teoremas matemáticos.

1956: Se inventa el termino Inteligencia Artificial por John McCarthy, Marvin Minsky y Claude Shannon en la Conferencia de Dartmouth.

1957: Newell y Simon continúan su trabajo con el desarrollo del General Problem Solver (GPS), el cual era un sistema orientado a la resolución de problemas.

1958: John McCarthy desarrolla en el Instituto de Tecnología de Massachusetts (MIT) LISP. Su nombre deriva del LISt Processor. LISP fue el primer lenguaje para procesamiento simbólico.

1959: Rosenblatt introduce el Perceptrón.

1959-1961: Robert K. Lindsay desarrolla “Sad Sam”, un programa para la lectura de oraciones en inglés y la inferencia de conclusiones a partir de su interpretación.

1963: Quillian desarrolla las redes semánticas como modelo de representación del conocimiento.

1964: Bertrand Raphael construye el sistema SIR (Semantic Information Retrieval) el cual era capaz de inferir conocimiento basado en información que se le suministra. Bobrow desarrolla STUDENT.

1965: Aparecen los sistemas expertos que predicen la probabilidad de una solución bajo un conjunto de condiciones.

1968: Marvin Minsky publica Semantic Information Processing.

1968: Seymour Papert, Danny Bobrow y Wally Feurzeig desarrollan el lenguaje de programación LOGO.

1969: Alan Kay desarrolla el lenguaje Smaltalk en Xerox PARC; se publica en 1980.

1968-1970: Terry Winograd desarrolla el sistema SHRDLU, que permitía interrogar y dar órdenes a un robot que se movía dentro de un mundo de bloques.
1973: Alain Colmenauer y su equipo de investigación en la Universidad de Aix-Marseille crean PROLOG (del francés PROgrammation en LOGique) un lenguaje de programación ampliamente utilizado en IA.
1973: Shank y Abelson desarrollan los guiones o *scripts*, pilares de muchas técnicas actuales en Inteligencia Artificial y la informática en general.
1974: Edward Shortlife escribe su tesis con MYCIN, uno de los sistemas expertos más conocidos que asistió a médicos en el diagnóstico y tratamiento de infecciones en la sangre.
1970-1980: Crece el uso de sistemas expertos.
1981: Kazuhiro Fuchi anuncia el proyecto japones de la quinta generación de ordenadores.
1986: McClelland y Rumberhart publican Parallel Distributed Processing (redes neuron ales).
1988: Se establecen los lenguajes orientados a objetos.
1997: Gari Kaspárov, campeón mundial de ajedrez, pierde ante el ordenador autónomo Deep Blue.
2006: Se celebro el aniversario con el Congreso en español 50 años de Inteligencia Artificial – Campus Multidisciplinar en Percepción e Inteligencia 2006.
2009: Se desarrollan sistemas inteligentes terapéuticos que permiten detectar emociones para poder interactuar con niños autistas.
2011: IBM desarrolla un superordenador llamado Watson, el cual gana una ronda de tres juegos seguidos de Jeopardy, venciendo a sus dos máximos campeones.
2016: Un programa informático gano cinco a cero al triple campeón de Europa de Go.
2018: Se lanza el primer televisor con Inteligencia Artificial por parte de LG Electronics con una plataforma denominada ThinQ.

De todos estos hechos, se podría asegurar que los siguientes ocho, de alguna manera u otra, ha dado forma a lo que ahora es la IA:

1956: El proyecto de investigación en Inteligencia Artificial en el verano, en Dartmouth, acuña el nombre de un nuevo campo relacionado con el quehacer de *software* habilidoso como los humanos.

1965: Joseph Weisenbaum en MIT crea Eliza, el primer Chatbot útil como psicoterapeuta.
1975: En Standford se desarrolla Meta-dendral, software útil para interpretar análisis químicos, aparece como publicación en una revista.
1987: Por primera vez una Van Mercedes Benz equipada con dos cámaras y varios ordenadores navega sola en terreno alemán alrededor de 20 km (55mph). Este proyecto fue dirigido por el ingeniero Ernst Dickmanns.
1997: El ordenador Deep Blue desarrollado por IBM derrota al campeón mundial del ajedrez, Garry Kasparov.
2004: El pentágono establece el proyecto Darpa Grand Challenge, una carrera para vehículos autónomos que da pie a esta industria.
2012: Los investigadores encuentran un nicho en lo que se conoce como aprendizaje profundo que abre un interés muy fuerte en IA; se muestra que sus ideas pueden hacer el reconocimiento del habla y las imágenes de modo más preciso.
2014: Alpha Go, creado por Google y DeepMind, derrota al campeón del mundo del entonces muy difícil juego del Go.

4.7.2 Ramas de la IA

Existen varios elementos que componen la ciencia de la IA, dentro de los cuales se pueden encontrar tres grandes ramas:

1. Lógica Difusa
2. Redes Neuronales Artificiales
3. Algoritmos Genéticos

Cada una consta de características especiales, así como de una función específica. En esta línea de investigación, nos centraremos principalmente en las Redes Neuronales Artificiales.

Aplicaciones comerciales de la IA

La IA ha sido usada en un amplio número de campos como la robótica, la comprensión y traducción de lenguajes, aprendizaje de palabras, etc. Los principales y más destacados campos donde se puede encontrar una notoria evolución de la IA son:

- Ciencias de la computación
- Finanzas
- Hospitales y medicina

- Industria pesada
- Servicio de atención al cliente
- Transportación
- Juegos

Un estudio llevado a cabo por la empresa Gartner en 2016 muestra que, en el pasado año 2020, al menos un 30% de las compañías comenzaron a usar IA y CD en al menos un fragmento de sus procesos de venta. Algunas áreas de oportunidad de la IA y la CD son:

- Asistentes personales
- Automóviles autónomos
- Banca y finanzas
- Edificios inteligentes
- Chatbots
- Cuidado de la salud
- Logística y cadenas de suministro
- Comercio electrónico (e-commerce)
- Diseño de ropa, estilos de zapatos, etcétera.
- Entretenimiento
- Electrodomésticos
- Telefonía móvil
- Turismo
- Servicio al cliente en línea

Algunos de ejemplos específicos de aplicación de la IA en los servicios tecnológicos son los siguientes:

Redes Sociales: En este caso se obtienen ejemplos de sistemas que:

1) Reciben fotos, las analizan y si encuentran personas hacer recomendaciones de etiquetado
2) Reciben como entrada fotografías, las analizan reconociendo los objetos ahí presentes y hacen recomendaciones de objetos similares

3) Son capaces de identificar significado contextual de emojis, pudiendo entonces sugerir en forma automática emojis de respuestas
4) Son capaces de seguir movimientos faciales y adicionar efectos animados o mascaras digitales cuando dichos rostros son usados por doquier en el mundo

Soluciones futuras: conversación con chatbots inteligentes. Estos chatbots deberán incorporar poderosos:

1) Procesadores de lenguaje natural
2) Reconocedores de rostros
3) Analizadores de expresiones faciales

Compras en línea: Muchas de las compras que se realizan actualmente son a través de internet. Incluye buscadores, recomen dadores, detectores de fraudes, etcétera. Soluciones futuras: sistema de compra completamente personalizados.

Desplazamiento: De acuerdo con un estudio en el año 2015 por parte del Instituto de Transporte de Texas y la Universidad de Texas A&M, los tiempos de desplazamiento en EUA han aumentado drásticamente, año con año. Esto ocasiono una perdida estimada de 160 billones de dólares en la productividad en el año 2014. Los sistemas de predicción que utilizan GPS, los sistemas de transporte compartido y los sistemas de autopiloto en los aviones ayudan a aminorar el problema. Las soluciones planteadas a futuro son las siguientes:

1) Vehículos autónomos
2) Transporte compartido optimizado
3) Sistemas de semáforos inteligentes (IOT)

4.7.3 Retos actuales para la IA

Aunque ya se cuenta con un mejor poder de cómputo (musculo), mejores técnicas para el manejo de grandes cantidades de datos para el aprendizaje de máquinas (aprendizaje profundo), todavía hay mucho camino que recorrer en materia de:

- Datos: La IA necesita miles de veces más datos que los requeridos por el cerebro humano para comprender conceptos y características. La capacidad de las máquinas para ver, entender e interactuar con el mundo está creciendo a un ritmo acelerado, apoyado en el volumen de datos que les ayuda a aprender y entender aún más rápida. Cada año la cantidad de datos que producimos se duplica. En la próxima década habrá aproximadamente 150 mil millones de sensores conectados a la red, equivalente a más de veinte veces la población de la Tierra. En este sentido, "Big Data" ha sido y será un gran aliado de la IA para procesar esta cantidad cada vez más grande de información y volverla útil.

- Inteligencia Artificial multitarea: Una vez que una máquina de IA es entrenada, es muy efectiva para tareas como el reconocimiento facial o de voz. Solo son capaces de realizar tareas específicas. Actualmente, no hay maquinas inteligentes capaces de cambiar de una tarea a otra. Esto es uno de los retos actuales de la IA
- Hardware: A pesar de la gran capacidad de procesamiento de han alcanzado los ordenadores actuales, toda la información que se encuentra disponible, así como sus técnicas de procesamiento y análisis desarrolladas a lo largo de los años, la IA como tecnología aún se encuentra limitada por el *hardware*. La infraestructura necesaria para experimentar y diseñar con la IA es todavía escasa y costosa. Se necesitan equipos cada vez más potentes para tener mejores resultados. Posibles soluciones a esta problemática involucran el desarrollo y puesta en operación de nuevos paradigmas de cómputo, como son el masivamente paralelo, cuántico, DNA, óptico, bacteriano, atómico, etcétera.

4.8 Redes Neuronales

La tecnología neural trata de reproducir el proceso de solución de problemas del cerebro. Son un modelo computacional basado en las neuronas del sistema biológico, con la finalidad de reproducir, de forma aproximada, el comportamiento de estas, así como los humanos aplican el conocimiento ganado con la experiencia a nuevos problemas o situaciones, una red neuronal toma como ejemplos problemas resueltos para construir un sistema que toma decisiones y realiza clasificaciones. Los problemas adecuados para la solución neural son

aquellos que no tienen solución computacional precisa o que requieren algoritmos muy extensos como en el caso del reconocimiento de imágenes. Funciona con un elevado número de unidades interconectadas nombradas neuronas. Las neuronas se organizan en capas. Normalmente una red neuronal está compuesta por tres partes: la capa de entrada, con neuronas que representan los campos de entrada; una o más capas ocultas; y una capa de salida con una o más unidades que representan el resultado computado por la red. En la primera capa se introducen los datos de entrada, y los valores se van propagando desde cada neurona hasta las neuronas de la capa siguiente. Finalmente, la capa de salida devuelve un resultado. Las conexiones entre neuronas están representadas por un peso, que puede ser positivo o negativo dependiendo si una neurona activa o inhibe a otra (ver Figura XVIII).

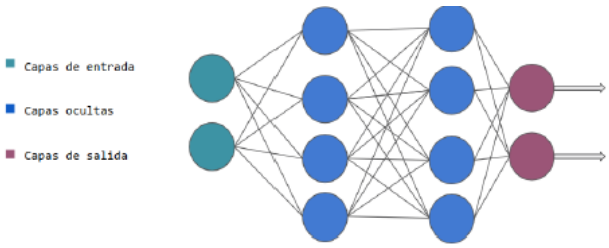


Fig. XVIII Red Neuronal Multicapa (Alejandro, 2017-2018)**.**

La red aprende a partir de un proceso de retroalimentación llamado Backpropagation. Consiste en comparar el resultado obtenido en la capa de salida con el resultado que se pretendía obtener. Con la diferencia entre ambos resultados se modifican los pesos de las capas ocultas hasta la capa de entrada. Después de varias iteraciones la red empieza a aprender y reduce la diferencia entre el resultado esperado y obtenido.

4.8.1 Historia de las redes neuronales

Alrededor de 1943 los investigadores Warren McCulloch y Walter Pitts propusieron el primer modelo simple de la neurona. En las décadas de los cincuenta y setenta, el movimiento en redes neuronales fue liderado por B. Widrow y M. E. Hoof., quienes trabajaron con una maquina llamada *Adeline* (Adaptive Linear Element). Otro pionero fue el psicólogo Frank Rosenblatt de la Universidad de Corell. En 1959, Rosenblatt construyo una maquina neural simple que llamo *perceptrón*. Este tenía una matriz con 400 fotoceldas que se conectaban aleatoriamente a 512 unidades tipo neurona. Cuando se representaba un patrón a las unidades

sensorias, estas enviaban una señal a un banco de neuronas que indicaba la categoría del patrón. El perceptrón de Rosenblatt reconoció todas las letras del alfabeto.

Al final de los años setenta Minsky y Papert demostraron que los *perceptrones* eran incapaces de hacer tareas simples tales como sintetizar la función lógica XOR. Las matemáticas del libro *Peceptrons* eran indiscutibles y su tono dio el mensaje que los perceptrones eran un camino sin salida. Uno de los científicos que continúo trabajando durante años oscuros de las redes neurales fue Stephen Grossberg, ahora director del Centro para Sistemas Adoptivos de la Universidad de Boston. Grossberg junto con Gail Carpenter de la Universidad de Northeastern han propuesto un modelo de red neural llamado ART (Adaptive Resonance Theory). Otros investigadores que trabajaron durante los años setenta fueron Teuvo Kohonen, de la Universidad de Helsinki, y Jim Anderson, de la Universidad de Brown, que trabajo con alternativas de semillas de conexionismo y junto con Geoff Hinto, quien presento trabajos matemáticos y de aplicación de redes neuronales organizaron el primer encuentro neoconexionista en 1979.

En 1986 McClelland y Rumelhart publicaron un libro en dos volúmenes titulado: *Parallel Distributed Processing: Explorations in the Microstruture of Cognition*. Este libro se considera un clásico en el área de redes neurales y se puede decir que su aparición significo un nuevo impulso a la investigación en sistemas neurales al mostrar las ventajas y desventajas de las redes neurales artificiales (RNA).

Algunas de las ventajas de las RNA frente a otros sistemas de procesamiento de información son:

- Las RNA pueden sintetizar algoritmos a través de un proceso de aprendizaje.
- Para utilizar la tecnología neural no es necesario conocer los detalles matemáticos. Solo se requiere estar familiarizado con los datos del trabajo.
- La solución de problemas no lineales es uno de los fuertes de las RNA.
- Las RNA son robustas, pueden fallar algunos elementos de procesamiento, pero la red continúa trabajando; esto es contrario a lo que sucede en programación tradicional.

Las desventajas de las redes neurales son:

- Las RNA se deben entrenar para cada problema. Además, es necesario realizar múltiples pruebas para determinar la arquitectura adecuada. El entrenamiento es largo y puede consumir varias horas de la computadora (CPU).
- Debido a que las redes se entrenan en lugar de programarlas, estas necesitan muchos datos.
- Las RNA representan un aspecto complejo para un observador externo que desee realizar cambios. Para añadir nuevo conocimiento es necesario cambiar las iteraciones entre muchas unidades para que su efecto unificado sintetice este conocimiento. Para un problema de tamaño considerable es imposible hacer esto manualmente, por lo tanto, una red con representación distribuida debe emplear algún esquema de aprendizaje.

Las redes neurales se basan en generalizar información extraída de datos experimentales, tablas bibliográficas o bases de datos, los cuales se determinan por expertos humanos. Dichas redes neurales toman en cuentan las entradas (corriente, voltaje) y como salidas las señales del sistema (velocidad, temperatura, torque). La red neural utilizada es una red multicapa de diez neuronas en la capa de entrada, diez neuronas en la capa oculta y cinco neuronas en la capa de salida. Por lo tanto, se tienen 250 pesos ajustables mediante un control retroalimentado o de lazo cerrado. En la siguiente figura se presenta un diagrama de una red neural, los parámetros de inicialización se obtuvieron mediante un conjunto de datos experimentales y una base de datos.

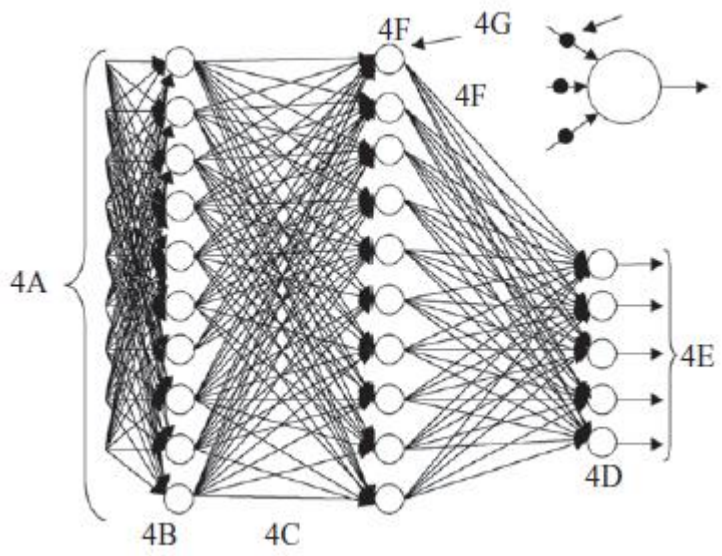


Fig. IXX. Esquema de la red neural multicapas 10-10-5 para el controlador inteligente. Entradas (4A): representa cualquier variable. Capa de entradas(4B), capa oculta(4C) y capa de salidas(4D). Salidas(4E): representa también cualquier variable de interés para el usuario. Los pesos entre cada neurona (4F) están representados por un punto negro(4G) (Ponce, 2010)**.**

El entrenamiento está basado en el algoritmo de "Retro propagación del error" por el método del gradiente descendiente, en donde los pesos se actualizan mediante el uso de un conjunto ordenado de entras y salidas deseadas y la comparación entre dicha salida y la salida real de la red neural. También se utiliza para el entrenamiento otra metodología alterna que es el "perceptrón". Es un clasificador de forma binaria: solo existe la posibilidad de ser parte de un grupo A o B, funciona con sistemas lineales.

4.8.2 Perceptrón

Es un tipo de red neural artificial. También puede entenderse como perceptrón la neurona artificial y unidad básica de inferencia en forma de discriminador lineal. Este consiste en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos inicialmente en forma aleatoria. En una fase en la que este aprende, la entrada se compra con un patrón preestablecido, la salida de la red es uno (1); en caso contrario la salida es cero (0). El perceptrón es un dispositivo que, en su configuración inicial, no está en capacidad de distinguir patrones de entrada muy complejos, sin embargo, mediante un proceso de aprendizaje es apto para adquirir esta capacidad. En esencia, el entrenamiento implica un

proceso de refuerzo a través del cual los pesos que codifican la sinapsis se incrementan o se disminuye. La red tipo perceptrón fue inventada por el psicólogo Frank Rosenblatt en el año 1957. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general, sin entrar en mayores detalles con respecto a condiciones específicas y desconocidas para organismos biológicos concretos.

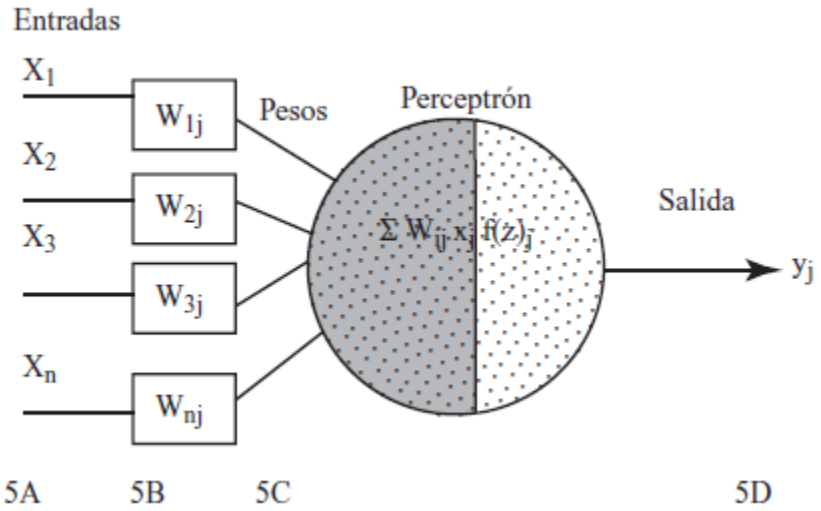


Fig. XX. Estructura del perceptrón, la más simple en las RNA. Es un discriminador binario lineal y puede ser entrenado para mejorar su desempeño. Las entradas de la neurona (5A), los pesos aleatorios (5B), la sumatoria de la multiplicación de los pesos por sus respectivas entradas (5C) y la salida que es el cálculo de todos los pesos y sus entradas (5D) (Ponce, 2010)**.**

En la figura anterior, se presenta una neurona "artificial", la cual intenta modelar el comportamiento de la neurona biológica. Aquí el cuerpo de la neurona se representa como un sumador lineal de los estímulos externos z_j seguida de una función no lineal $y_j = f(z_j)$. La función $f(z_j)$es llamada la función de activación y es la función que utiliza la suma de estímulos para determinar la actividad de salida de la neurona. Este modelo es la base de la mayoría de las arquitecturas de las RNA que se interconectan entre sí. Las neuronas emplean funciones de activación diferentes según la aplicación. Algunas veces son funcionales lineales, otras son funciones sigmoidales (por ejemplo, la $\tan x$) y otras son funciones de umbral de disparo. La eficiencia sináptica se representa por factores de peso de interconexión w_{ij}, desde la neurona i hasta la neurona j.

Los pesos pueden ser positivos (excitación) o negativos (inhibición). Los pesos junto con las funciones *f(z)* dictan la operación de la red neural. Normalmente las funciones no se modifican de tal manera que el estado de la red neural depende del valor de los factores de peso (sinapsis) que se aplica a los estímulos de la neurona. En un perceptrón, cada entrada se multiplica por el peso *w* correspondiente y los resultados se suman, siendo evaluados contra el valor de umbral; si el resultado es mayor al mismo, el perceptrón se activa.

El perceptrón solo es capaz de resolver funciones definidas por dos dimensiones. Un ejemplo de una función que no puede ser resuelta es el operador lógico. El entrenamiento de un perceptrón es por medio de la regla de aprendizaje delta:

- Para cada peso *w* se realiza un ajuste *dw* según la regla:

$$dw = \eta(x - Y)X$$

Donde η es la razón de aprendizaje, *y* el valor deseado, *Y* el valor obtenido y *X* la entrada aplicada al perceptrón.

4.8.2.1 Perceptrón de Rosenblatt

El perceptrón original de Rosenblatt consta de tres capas: Un de unidades de sensado (unidades tipo S), una de asociación (unidades tipo A) y una de respuesta (unidades tipo R). la retina del perceptrón es un arreglo de elementos de sensado (fotoceldas). Una unidad tipo S emite un "1" si es excitada, un "0" si no.

En el modelo original, un número de unidades de retina (retinales) seleccionado al azar era conectado a la capa de asociación (unidades tipo A). también existían muchas conexiones entre las unidades tipo A y tipo R, incluso de retroalimentación entre las del tipo R y las del tipo A. Esto, por supuesto, pudiera introducir dinámicas no deseadas en la operación del perceptrón. Una versión simplificada sin conexiones laterales y conexiones de retroalimentación se muestra en la Fig. XXI.

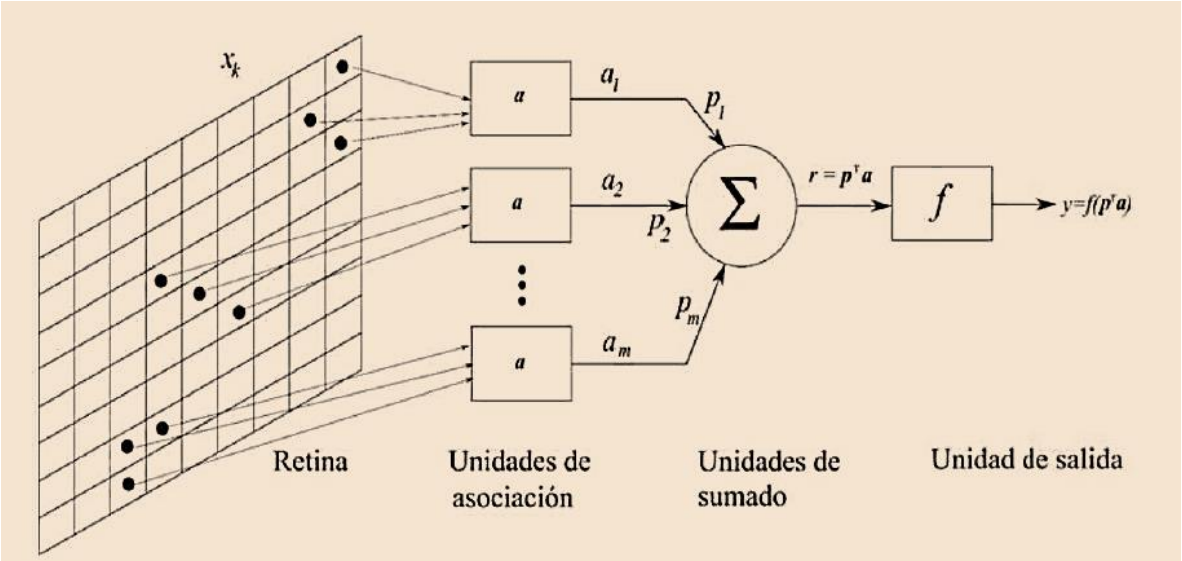


Fig. XXI. versión simplificada del perceptrón original (Humberto S. Azuela, 2020)**.**

De acuerdo a la figura, cada unidad tipo A funciona como sigue. La unidad recibe a su entrada *n* valores del tipo x_k, *generalmente al azar.* La unidad A calcula entonces una suma ponderada sobre los valores x_k:

$$\alpha_i = \sum_{k=1}^{n} \beta_k x_k$$

Los pesos β_k pueden tomar valores de +1 o -1, y son asignados en forma aleatoria. La suma $\sum_{k=1}^{n} \beta_k x_k$ es comparada con un umbral *u*. la valida *ai* de la i- ésima unidad A viene dada por la siguiente ecuación:

$$\alpha_i = \begin{cases} 1\ si\ \sum_{k=1}^{n} \beta_k x_k \geq u \\ 0\ si\ \sum_{k=1}^{n} \beta_k x_k < u \end{cases}$$

La salida binaria de la i-ésima unidad A, con *i=1, 2, ..., m* es multiplicada por un peso *pi*, y una suma ponderada de las *m* salidas ponderadas es obtenida por la unidad de suma. Cada peso *pi* puede tomar ser negativo, positivo o cero. La salida de la neurona es binaria: {0, 1} o {-1, 1}, dependiendo de un umbral θ, que normalmente es puesto en cero:

$$y = \sum_{i=1}^{m} \rho_i \delta_i$$

4.8.2.2 Perceptrón Estándar

La estructura del perceptrón estándar, normalmente usado en la literatura actual, se muestra en la Fig. XXII; para el cual la salida toma la forma de limite simétrico (fig. XXII (a), sin vías y XXII (b), con vías). Permite separar conjuntos de patrones linealmente separables en dos clases: *C1 y C2.*

La respuesta del perceptrón puede ser expresada como sigue:

$$y = f(r)$$

Donde, como ya se vio:

$$r = \sum_{i=1}^{n} p_i x_i = \boldsymbol{p}^T x,$$

Con **p**=$[p_1, p_2, \dots, p_n]^T$ y $x = [x_1, x_2, \dots, x_n]^T$;

$$r = \sum_{i=1}^{n+1} p_i x_i = \acute{\boldsymbol{p}}^T \acute{x},$$

Con $\acute{\boldsymbol{p}}$=$[p_1, p_2, \dots, p_n, p_{n+1} = \theta]^T$ y $\acute{x} = [x_1, x_2, \dots, x_n, x_{n+1} = -1]^T$;

Se puede ver que cuando:

1) $r \geq \theta$ para el caso de la primer sumatoria o $r \geq 0$ para el caso de la segunda sumatoria, la función de activación del perceptrón ocasiona que la salida y= +1, indicando que el patrón de entrada ***x*** debe ser clasificado en la clase C2.
2) $r < \theta$ para el caso de la primer sumatoria o $r < 0$ para el caso de la segunda, la función de activación del perceptrón ocasiona que la salida y=-1, indicando que el patrón de entrada ***x*** debe ser clasificado en la clase C1.

Un perceptrón como el que se muestra a continuación, se puede usar en la clasificación de patrones, sus pesos deben ser ajustados. Se puede emplear la misma regla de entrenamiento utilizada para el ajuste de parámetros libres de la UUL (máquina de separación lineal).

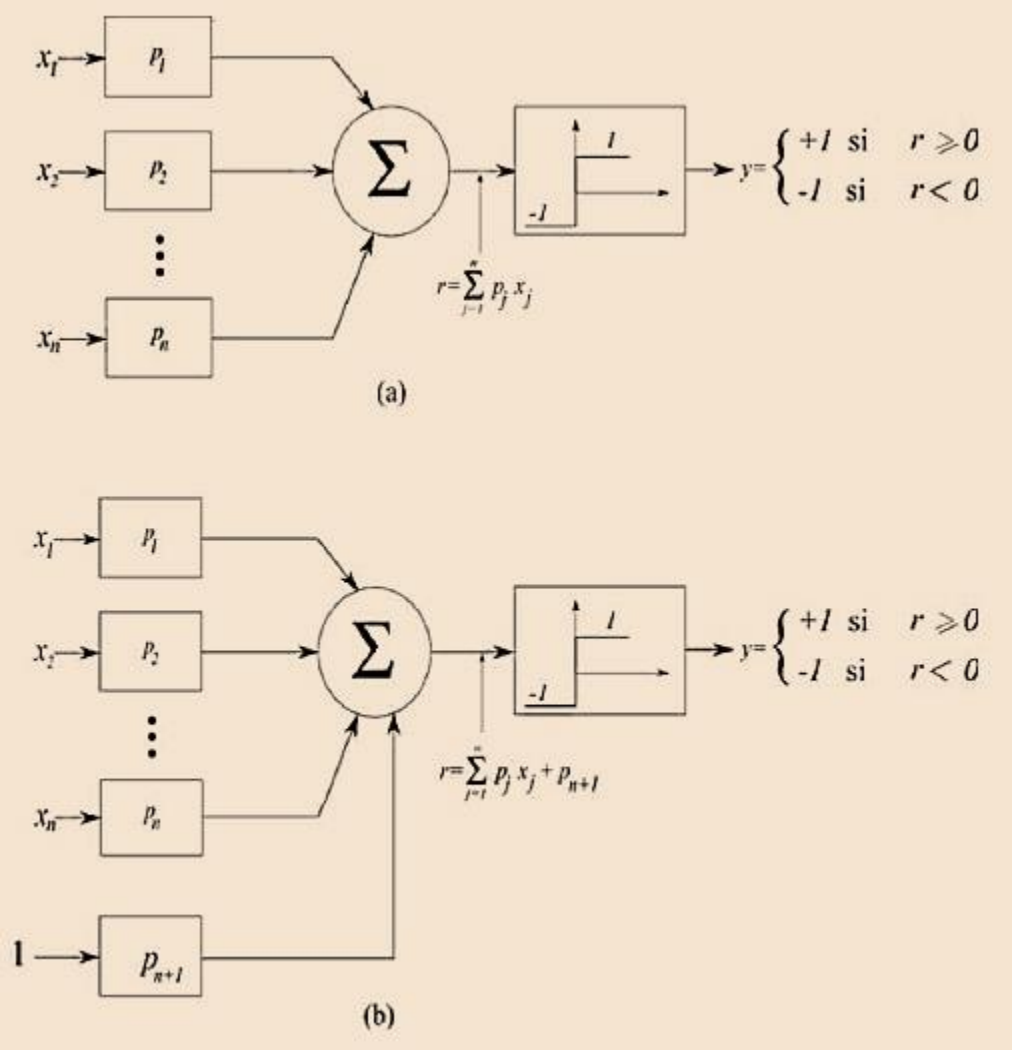


Fig. XXII. Representación esquemática del perceptrón para dos clases. (a) sin vías o desplazamiento, y (b) con vías (Humberto S. Azuela, 2020)**.**

De todo esto, se puede ver por medio de una UUL o resolver a través de un perceptrón, en principio, cualquier problema que involucre la separación de un conjunto de patrones en dos clases. Sin embargo, como se ha mencionado, si el conjunto de patrones no presenta separación lineal, entonces el perceptrón, al igual que la UUL, no podrá separarlos.

4.8.3 Redes Neuronales en la Agricultura de Precisión

La agricultura de precisión se define como “un conjunto de técnicas que permiten la gestión localizada, y su éxito depende de tres elementos: información, tecnología y gestión”. Con la creciente adopción de diferentes técnicas de manejo del suelo, tales como fertilizantes a tasas variables y la posibilidad de la adopción de máquinas específicas en la agricultura de precisión, es necesario establecer sitios específicos de gestión *diferentes áreas de gestión*. La gestión eficaz de manejo por sitio específico requiere una comprensión de los suelos y de los factores ambientales que afectan a la variabilidad de la producción agrícola; además de ser necesaria para evaluar las técnicas utilizadas para definir estas relaciones.

El origen de las técnicas surge de entender cómo las propiedades del suelo afectan a la producción, así como la posibilidad de utilizar estos en la predicción del rendimiento de la cosecha, con el fin de definir la gestión de sitios específicos, planificación de la inversión, la predicción de la productividad y el beneficio.

Estos modelos se pueden basar en redes de sistemas computacionales paralelas, que consisten en unidades de procesamiento simples, también llamados neuronas artificiales conectadas entre sí de una manera particular para realizar una tarea.

RNAs han demostrado un alto rendimiento debido a factores tales como ser distribuidos o en paralelo y la estructura robusta conocidas como capas; la eficiencia en el aprendizaje y la generalización, lo que las hace capaces de resolver problemas complejos; son tolerantes a los valores típicos o "atípicos"; pueden modelar diferentes variables y sus relaciones no lineales; y, por último, permitir el modelado con variables categóricas.

Actualmente, estas técnicas se desarrollan en una configuración de idoneidad y de parámetros para diferentes situaciones como: las funciones tapper o adelgazamiento del árbol, el modelado diametral, el modelo de crecimiento para el corte de eucalipto, los estudios de las características del suelo; recientemente, para el trigo con la predicción de rendimiento de grano.

Sin embargo, en la productividad estimada de granos en base a parámetros físicos y químicos del suelo, para establecer la gestión de sitios específicos, el comportamiento de las RNAs no

se ha explorado lo suficiente. No obstante, existe la posibilidad de establecer modelos bioma temáticos capaces de predecir el rendimiento del cultivo en "segunda cosecha" a través de las propiedades del suelo utilizando modelos RNA bioma temáticos o regresión múltiple.

Un método comúnmente ocupado para evaluar el grado de relación entre las variables involucradas en el proceso de modelado es el análisis de correlación de Pearson. Secuencialmente, los modelos de diferentes categorías de configuración *redes neuronales y regresión múltiple* se pueden ajustar para representar estos modelos. Por ejemplo, para estimar la productividad se puede utilizar el método de regresión lineal múltiple de mínimos cuadrados ordinarios:

$$Y = \beta_0 + \beta_1 M0 + \beta_2 xCTC + \beta_3 xV(\%) + \beta_4 xTA$$

Donde Y es el rendimiento promedio de la cosecha (kg) en el periodo a evaluar; MO contenido de sólidos, es decir, el suelo orgánico (mg); CTC Capacidad de intercambio Catiónicos (mmol); V (%) es la saturación de base; RT se define como la resistencia de la arcilla (mg); βi = Estimadores de parámetros a ser ajustados mediante i = 0,1,2,3 y 4.

Las redes neuronales consideran las mismas variables; sin embargo, las RANs utilizan la inteligencia artificial para resolver los problemas de ajustes, que se forman por elementos de procesamiento simples, estas se activan mediante una función (función activación), para conseguir una respuesta única. En estas neuronas artificiales, la información de la unidad de procesamiento consiste en entradas de "n" x1, x2... Xn (Dendritas) y una salida (axón); las entradas se asocian con los pesos W1, W2..., Wn representando las sinapsis. Este modelo lo podemos representar de la siguiente manera:

$$Y_k = \varphi(V_k)$$

Dónde: Y_k= Salida de la neurona artificial; función φ = activación; V_k = Resultado del combinador.

Para utilizar la red, optamos por la técnica de perceptrón multicapa donde inicialmente, los pesos de todas las redes se generan al azar. Secuencialmente, esta actualización de valor individual evoluciona durante el proceso de aprendizaje basado en la función del error.

Las estimaciones de los rendimientos de cosecha se simularán con las posibles combinaciones de variables de entrada, un total de cuatro combinaciones a la variable de respuesta. La función de activación utilizada en este método es la sigmoidea ya que es la más común en desarrollo de las redes neuronales artificiales.

4.8.4 Redes Convolucionales

La clasificación de imágenes a partir de una red neuronal como las vistas en la sección anterior puede ser un problema debido a la carga computacional que esto supone. Como se comenta, en las redes neuronales todas sus neuronas están totalmente conectadas entre capas. Si por ejemplo se dispone de una imagen con unas dimensiones de 640 x 640 píxeles, esta hace un total de 409.600 datos de entrada que estarán conectados a cada una de las neuronas de la primera capa. Esto produce un gran número de cálculos que ralentizará el proceso de entrenamiento.

Para mitigar este problema se presentan las redes neuronales convolucionales. Las redes neuronales convolucionales están formadas por diferentes capas (ver Figura XXIII), donde las primeras capas se encargan de extraer características como los bordes o esquinas de una imagen. Una vez detectados estos bordes son utilizados para detectar formas en las capas posteriores. Cuando se detectan las formas se procede a detectar las características de alto nivel como puede ser una rueda en una imagen donde aparece un coche. Las últimas capas están totalmente conectadas y son las encargadas de dar una predicción a partir de estas últimas características extraídas.

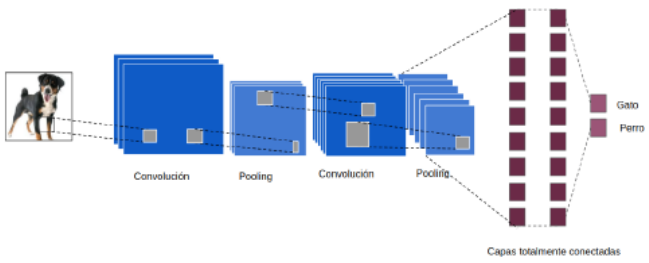


Fig. XXIII. Red Neuronal Convolucional (Alejandro O. R., 2017-2018)**.**

Una red neuronal puede representarse como un grafo dirigido con las siguientes propiedades

- A cada nodo j se le asocia una variable de estado xj.
- A cada conexión (i, j), entre los nodos i y j, se le asocia un peso $wij \in \mathbb{R}$
- En muchos casos, a cada nodo se le asocia un umbral de disparo θj
- Para todo nodo j, se define una función $fj\ (xi, wij, j)$, que depende del estado de todos los nodos unidos a él, de los pesos de sus conexiones y del umbral de activación para proporcionar un nuevo estado.

Considerando el lenguaje habitual de los grafos, pueden establecerse las siguientes equivalencias:

- Un nodo representa una neurona.
- Una conexión representa una sinapsis.
- Una neurona de entrada es aquella sin conexiones entrantes.
- Una neurona tanto con entradas como salidas, se denomina neurona oculta.
- Una neurona de salida es la que no presenta conexiones salientes.

El número de neuronas por cada capa está determinado por las propiedades del problema a resolver. Generalmente, el número de neuronas de la capa de entrada coincide con la cantidad de entradas individuales de la red, y el número de neuronas de la capa de salida es generalmente el número de clases en las que separa la solución. El número de neuronas de la capa intermedia no está sujeta a ningún parámetro.

Una red neuronal para la clasificación de imágenes, generalmente consta de tres capas. Una capa de entrada, donde por cada clase descriptora de los rasgos de la imagen (textura, color, forma, etc.) se tiene una neurona. Una capa intermedia u oculta, que es donde se efectúa el aprendizaje de la red y una capa de salida, generalmente con la misma cantidad de neuronas que de clases que se quieren clasificar con la red.

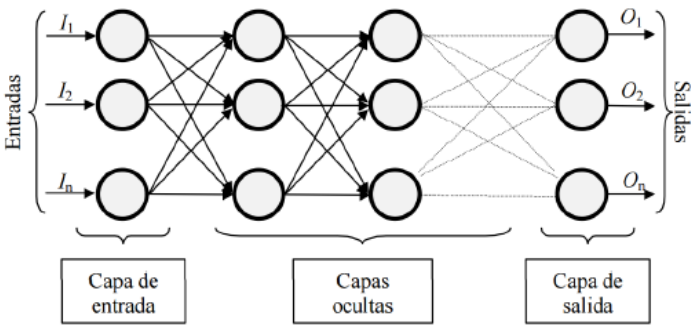


Fig. XXIV. Estructura de una Red Neuronal para la clasificación de Imagen (Andrés F. Montenegro B., 2015)**.**

Capitulo 5

Metodología

- 5.1 Desarrollo Tipo Cascada
- 5.2 Diseño de la Red Neuronal y Base de Datos.
- 5.3 Implementación de la Metodología.
- 5.4 Tecnologías Empleadas.
- 5.5 Implementación.

5.1 DESARROLLO TIPO CASCADA

Durante el desarrollo de y proceso de investigación y la programación de la red neuronal, el modelo en cascada fue fundamental para poder llevar a cabo el proyecto, ya que este es un proceso de desarrollo secuencial de etapas que se ejecutan una tras otra, debido a esto, se lleva un orden en específico para el desarrollo del sistema y un orden para la realización de la investigación , con el cual primero se empezó por la obtención de la problemática a resolver, seguido de una exhaustiva investigación para poder dar una solución al problema, luego, llevar a cabo el diseño del sistema y la codificación de este, terminada la codificación se lleva a cabo la implementación y verificación del sistema, para que a través de esto se pueda a llevar a cabo modificaciones en el sistema, una vez aprobado el funcionamiento de la red neuronal en las pruebas, se lleva a cabo el mantenimiento del sistema para futuras actualizaciones o modificaciones que se puedan presentar.

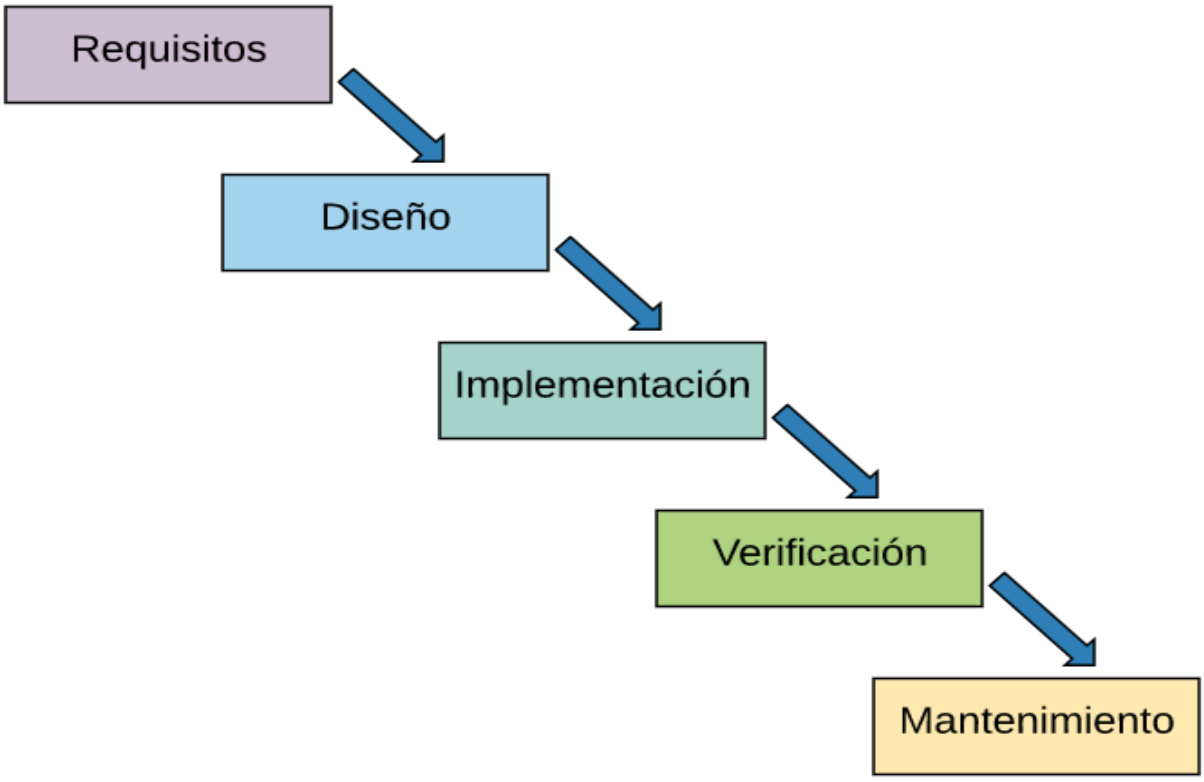


Fig. XXV. Etapas del Modelo en Cascada.

El modelo de desarrollo en cascada sigue una serie de etapas de forma sucesiva, la etapa siguiente empieza cuando termina la etapa anterior. Las fases que componen el modelo son las siguientes:

- Requisitos del software

En esta fase se hace un análisis de las necesidades de los diferentes cultivos de riegos existentes en el ámbito de agricultura, para determinar las características del software a desarrollar, y se especifica todo lo que debe hacer el sistema sin entrar en detalles técnicos.

En esta es la etapa en la que se lleva a cabo una descripción de los requisitos del software, para ello, se lleva a cabo una investigación de campo en los riegos de cultivos cercanos, enfocado en una planta en común, como lo es el maíz, para observar las diferentes técnicas de detección de plagas que se utilizan, una vez realizada la investigación se diseña lo que el producto final deberá hacer. Disponer de una especificación de los requisitos permite estimar de forma rigurosa las necesidades del software antes de su diseño. Además, permite tener una base a partir de la cual estimar el coste del producto, los riesgos y los plazos.

En el documento en el que se especifican los requisitos, se establece una lista de los requerimientos acordados. Esto se consigue teniendo una lista detallada de los requisitos, y con una comunicación fluida según las revisiones propuestas en el laboratorio para evaluar a futuro el avance progresivo hasta la etapa final de desarrollo.

- Diseño

En esta etapa se describe la estructura interna del software, y las relaciones entre las entidades que lo componen.

Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe

hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

- Implementación

En esta fase se programan los requisitos especificados haciendo uso de las estructuras de datos diseñadas en la fase anterior. La programación es el proceso

que lleva de la formulación de un problema de computación, a un programa que se ejecute produciendo los pasos necesarios para resolver dicho problema.

Al programar, se debe realizar actividades como el análisis de las condiciones, la creación de algoritmos, y la implementación de éstos en un lenguaje de programación específico.

Un algoritmo es un conjunto de instrucciones o reglas bien definidas y ordenadas que permiten llevar a cabo una actividad mediante pasos sucesivos.

- Verificación

Como su propio nombre indica, una vez se termina la fase de implementación se verifica que todos los componentes del sistema funcionen correctamente y cumplen con los requisitos.

El objetivo de las pruebas es el de obtener información de la calidad del software, y sirven para: encontrar defectos o bugs, aumentar la calidad del software, refinar el código previamente escrito sin miedo a romperlo o introducir nuevos bugs, etc.

- Instalación y mantenimiento

Una vez se han desarrollado todas las funcionalidades del software y se ha comprobado que funcionan correctamente, se inicia la fase de instalación y mantenimiento. Se instala la aplicación en el sistema y se comprueba que funcione correctamente en el entorno en que se va a utilizar.

A partir de ahora hay que asegurarse de que el software funcione y hay que destinar recursos a mantenerlo. El mantenimiento del software consiste en la modificación del producto después de haber sido entregado al cliente, ya sea para corregir errores o para mejorar el rendimiento o las características.

El propósito de esta fase es mantener el valor del software a través del tiempo. Esto puede hacerse añadiendo nuevos requisitos, corrigiendo errores, renovando el aspecto visual, mejorando la eficiencia o añadiendo nueva tecnología. El

periodo de mantenimiento puede durar años, por lo que es una fase clave del modelo en cascada.

Para llevar a cabo correctamente la fase de mantenimiento, se necesita trazar un plan de antemano que nos prepare para todos los escenarios que puedan producirse durante esta fase. Para evitar futuros conflictos con el cliente, en el plan hay que especificar cómo los usuarios solicitarán las modificaciones o la corrección de errores, hacer una estimación del coste de la modificación de funcionalidades o corrección de errores, quién se encargará del mantenimiento, durante cuánto tiempo se dará soporte al software, etc.

5.2 DISEÑO DE LA RED NEURONAL Y BASE DE DATOS

Este diseño se basa en una metodología diseñada para la identificación de plagas y enfermedades de los cultivos de riegos y en el esquema de entrenamiento de una red neuronal (Fig. XXVI), en específico de las plantas de maíz; donde se integran información de:

1.- Planta de Maíz,

2.-Color de hoja,

3.-Manchas Foliares,

4.-Color de la planta sana,

7.-Color de la planta enferma.

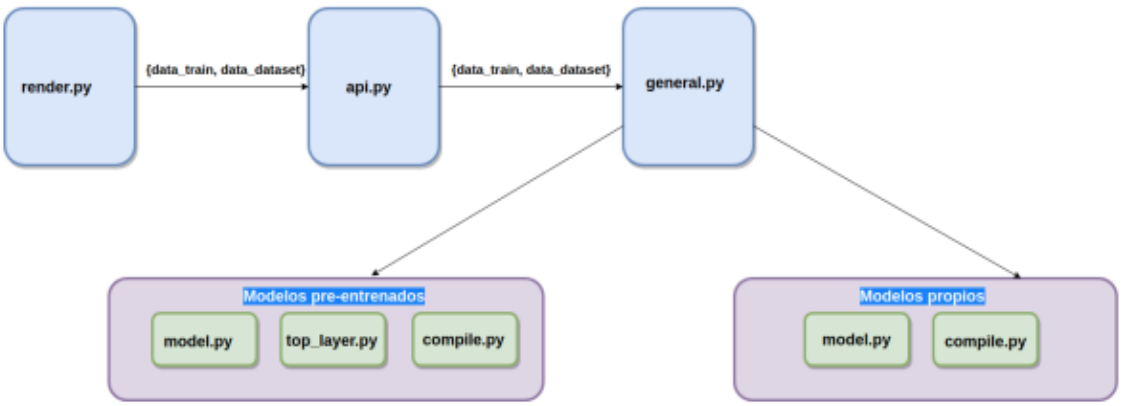


Fig. XXVI. Esquema de Entrenamiento de una Red Neuronal para la identificación y clasificación de imágenes (Alejandro O. R., 2017-2018)**.**

Los datos se procesan mediante imágenes a través de la programación en lenguaje Python y se codifican a codebooks para transformarlos a vectores de información; es decir, se realiza un proceso de identificación de datos a través de la introducción y recopilación de estos, que pasan a ser codificados con base a los algoritmos programados para integrarlos en una matriz binaria de datos, que se guardan en archivos denominados codebooks, los cuales llamaremos "pesos" y "modelo", los cuales, la red neuronal, lee y analiza para poder hacer una predicción de (fig.XXVII):

1.- Planta Enferma

2.- Planta Sana

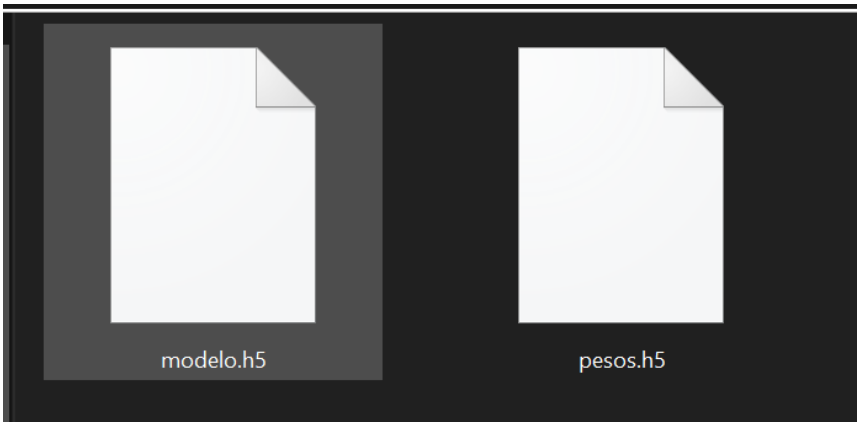


Fig. XXVII. Codebooks en los que se transforman los datos de la imagen para el entrenamiento de la red neuronal.

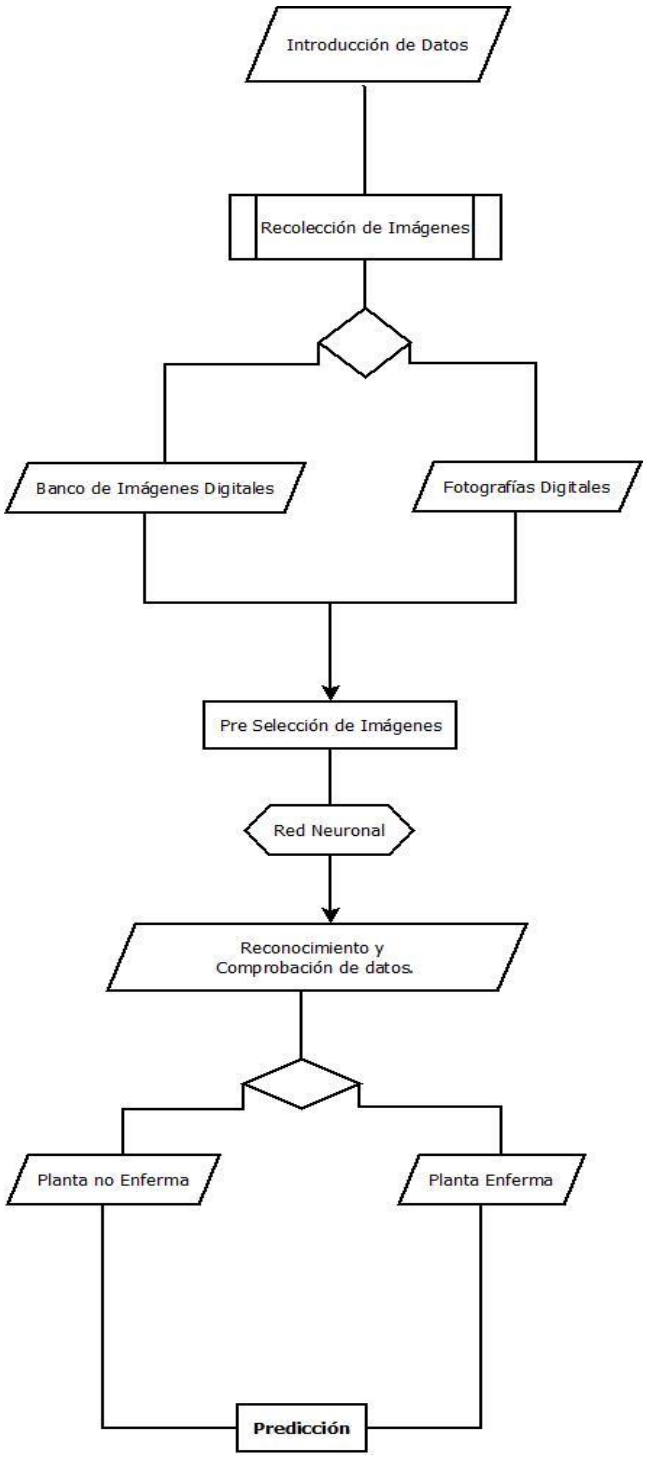


Fig. XXVIII. Diagrama de Flujo.

5.3 IMPLEMENTACIÓN DE LA METODOLOGÍA

En este breve apartado se menciona el paradigma para la implementación del proyecto. A partir de los requisitos y objetivos del proyecto y el uso de tecnologías que son mencionados a continuación, se decide utilizar un paradigma orientado por Fases, como especifica el método de cascada.

Para poder realizar la creación del sistema, se utilizó la programación dirigida por eventos; en donde la ejecución del programa va determinada por los eventos provocados por el usuario o el propio sistema. Es trabajo del desarrollador definir los eventos que manejan el programa y las acciones a realizar cuando se produce un evento.

En el desarrollo de la herramienta se han implementado diferentes eventos que son inicializados cuando empieza la ejecución del programa. Sin embargo, también se implementan eventos de forma dinámica. Se anticipa que se establece una comunicación entre diferentes tecnologías a consecuencia de la interacción del usuario con la interfaz. Estas tecnologías se dividen entre las que conforman la interfaz del usuario y las empleadas para trabajar con modelos de aprendizaje profundo. Aunque el usuario no interactúa directamente con las tecnologías externas de la herramienta (Tecnología Python), en la siguiente figura se presenta un esquema básico sobre como es el comportamiento de un sistema a través de eventos.

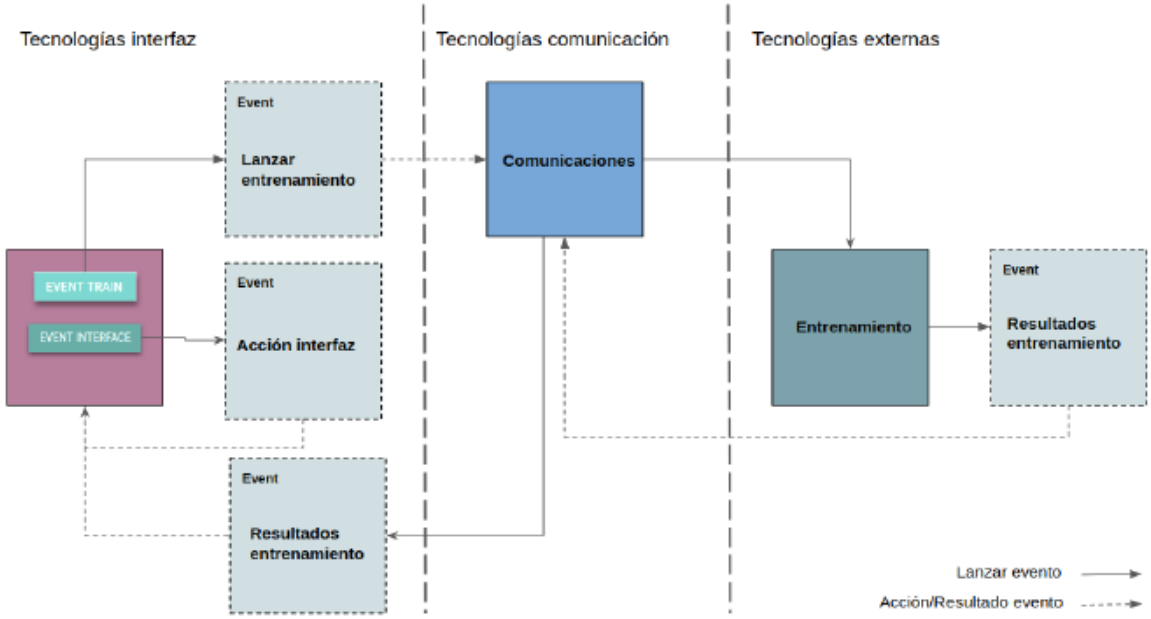


Fig.XXIX Esquema básico de la interacción eventos en la herramienta (Alejandro O. R., 2017-2018)**.**

.

5.4 TECNOLOGÍAS EMPLEADAS

A continuación, se exponen las principales características de las tecnologías utilizadas.

Python

Python es un lenguaje de programación multiparadigma soportando la orientación a objetos, programación imperativa y programación funcional. Es un lenguaje interpretado, usa tipo dinámico y es multiplataforma.

Puede distribuirse libremente ya que su interprete como la biblioteca estándar se encuentra de forma binaria y en código fuente en la página web de Python (https://www.python.org/). En la web se pueden encontrar también distribuciones y módulos desarrollados por terceros, así como documentación adicional. El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++.
Es un lenguaje que está ganando usuarios que se dedican al campo de la inteligencia artificial. El *framework* Keras utilizado para el entrenamiento de redes neuronales es una *API* integrada en Python, siendo esta la razón de la elección para la integración en el proyecto. Mediante

llamadas a procesos remotos se pueden establecer la comunicación con la herramienta desarrollada con Spyder (Anaconda) como IDE para la programación de Python.

TensorFlow

Es una biblioteca de software de código abierto para el aprendizaje automático. Fue creado y lanzado por Google en el año 2015 consiguiendo un gran éxito hasta el momento. Cuenta con una gran cantidad de colaboradores que lo sitúan como líder en el sector del aprendizaje profundo.

Se trata de una plataforma creada para la construcción y entrenamiento de redes neuronales, que permite detectar y descifrar patrones propios de los seres humanos. Puede realizar operaciones de cálculo tanto en CPU o GPU.

Para el entrenamiento de redes neuronales convolucionales ofrece un extenso soporte para trabajar con GPU tanto de equipos de escritorio como en servicios Cloud, por ejemplo: Google Colab. Para el presente proyecto se hace uso directo de Tensorflow debido a la integración de la biblioteca Keras.

Keras

Es una *API* de redes neuronales de alto nivel, escrita en Python y con posibilidad de integrarse con TensorFlow. Su desarrollo se basó en crear una *API* con la que realizar un experimento sobre una red fuese una tarea sencilla y rápida.

La facilidad de uso es uno de los principales objetivos, ofreciendo una *API* consistente y simple que minimiza el número de acciones que ha de realizar el usuario. Ofrece una documentación extensa con algunos ejemplos que pueden ser de gran ayuda para empezar a trabajar en el campo de las redes neuronales. Separa por módulos cada una de las partes de una red neuronal. Las capas de una red neuronal, la función de coste, la función de activación, la función de regularización es entre otros los módulos por los que esta formado. Se pueden crear módulos personalizados fácilmente, haciendo de Keras un sistema apto para investigaciones avanzadas.

Los modelos se escriben en código Python. La implementación en Python ofrece la ventaja a la hora de la depuración y extensión de los modelos. Gracias al desarrollo de librerías como Keras, Python está aumentando en el sentido de desarrolladores que trabajan con él. Se ha elegido Keras para integrarlo en el proyecto por la facilidad de la implementación a una red neuronal.

OpenCV

Es una biblioteca libre de visión por computadora desarrollada por Intel. Desde su aparición en 1999 ha sido utilizada en infinidad de aplicaciones. Esto se debe a que su publicación se da bajo licencia *Berkeley Software Distribution* (BSD), que permite que sea utilizado libremente para fines comerciales y de investigación con las condiciones expresadas.
Es multiplataforma, con versiones para distribuciones GNU/Linux, Mac OS X, Windows, Android y IOS. Tiene interfaces C++, Python y Java. Sus funciones abarcan grandes áreas en el proceso de visión como el reconocimiento de objetos, calibración de cámaras o visión robótica.
Otras funciones como redimensionamiento o recorte de imágenes también están incluidas en la biblioteca. Estas características junto la detección de objetos ha decantado la elección para la integración en el proyecto.

VGG16

Las redes VGG (*Visual Geometry Group*) son unas redes neuronales propuestas en 2014 para el reconocimiento de imágenes a gran escala. Todas tienen el mismo patrón de estructura, capas convolucionales de 64m 128, 256 y 512 filtros de dimensión 3. Las diferencias entre unas y otras residen en el número de capas totales, por ejemplo, la red VGG-13 tiene 13 capas en total, contando convolucionales y completamente conectadas.

Las redes VGG están diseñadas para recibir inputs de imágenes 224x224x3 y para dar como salidas una clasificación de 1000 clases. Se ha adaptado VGG-16 para el caso de regresión,

modificando la primera y la última capa para que acepte un input de 112x112x15 y se obtenga como output un solo valor.

Uno de los inconvenientes de este tipo de redes es el tiempo de entrenamiento y el tamaño del archivo de la red, siendo ser ambos elevados debido al número de parámetros involucrados, pero una de las principales ventajas que tiene el modelo VGG-16 es la capacidad de poder ser entrenada con pocas imágenes de entrada.

5.5 IMPLEMENTACIÓN

Como se ha ido viendo, para este proyecto de investigación aplicada, se implementa la metodología en Cascada para poder abarcar cada uno de los objetivos planteados, a continuación, se detallan las actividades, herramientas y resultados esperados en cada actividad.

Objetivo 1: Análisis de Imágenes en Base de Datos		
Actividades	**Herramientas**	**Resultados**
Identificar variables descriptoras de las plantas de maíz.	Revisión Documental	Sección con descripción de los atributos y variables a medir.
Elección de las enfermedades y/o plagas a analizar	Revisión Documental, Juntas con el tutor para determinar cuál es la opción más viable para trabajar.	Enfoque en la enfermedad “Mancha Foliar” para llevar a cabo la investigación.
Toma de Imágenes	Cámara Fotográfica, Banco de imágenes online.	Conjunto de imágenes de plantas de maíz y hojas de maíz por cada muestra a analizar.
Realizar análisis de Procesamiento digital de las imágenes recopiladas	Software Python	Mejora de imágenes, Nuevas imágenes para integrar al banco de datos, mejora de calidad de imágenes originales.

Objetivo 2: Identificar, obtener y recopilar imágenes descriptoras de las plantas de maíz con Mancha Foliar y plantas Sanas, para usarlos como datos de entrada del modelo.		
Actividades	**Herramientas**	**Resultados**
Revisión de métodos disponibles para detección de plagas	Revisión documental, Agricultores dedicados al cultivo de maíz	Experiencia en el campo laboral sobre como trabajar con el maíz, Capitulo descriptivo de los métodos utilizados de la agricultura tradicional.
Clasificación de muestras de plantas de maíz	Agricultores dedicados al cultivo de maíz.	Calificación de la calidad de las plantas según el método seleccionado.

Objetivo 3: Realizar evaluación tradicional de la planta de maíz, para poder analizar las metodologías utilizadas para la detección de plagas o enfermedades, las cuales serán usadas como salidas o etiquetas para el modelo.		
Actividades	**Herramientas**	**Resultados**
Uso de métodos tradicionales de selección de hojas de maíz.	Agricultores dedicados al cultivo de maíz, uso de la vista para la identificación de las hojas.	Clasificación de las hojas de maíz según el método seleccionado
Reporte de resultados	Anotaciones sobre el método de clasificación de las hojas	Reporte final sobre los principales síntomas de una planta de maíz contaminada con Mancha Foliar
Etiquetas de plantas contaminadas	Etiquetas, Varas para marcar territorio, Reporte Final	Clasificación final de las plantas contaminadas en un área real de trabajo

Objetivo 4: Implementación de Keras		
Actividades	**Herramientas**	**Resultados**
Investigación del método keras	Reportes, Tesis, Libros	Reporte de investigación sobre qué es y cómo funciona el método de Keras

Objetivo 5: Creación de Red Convolucional en base al Modelo VGG16		
Actividades	**Herramientas**	**Resultados**
Construcción de base de datos	Software Python	Base de datos con las variables medidas a cada muestra y los resultados de calidad evaluados.

Preparación de los datos para ingresar al modelo	Software Python	Métodos de Pre Procesamiento de Imágenes para mejora de calidad. Datos normalizados. División de datos entre conjunto de entrenamiento y prueba
Preparar imágenes tomadas	Software TensorFlow	Base de datos de imágenes de plantas y hojas de maíz etiquetadas listas para entrenar el modelo
Elección y entrenamiento de los algoritmos para la predicción de calidad	Software Python	Modelo Entrenado
Entrenamiento de red neuronal para el reconocimiento de imágenes.	Software Python, API Keras, TensorFlow.	Modelo entrenado, resultados de indicadores de desempeño del modelo.
Ajuste de parámetros del modelo Entrenado	Software Python, API Keras, TensorFlow.	Modelo ajustado con mejoras en los indicadores.

Objetivo 6: Entrenar, evaluar y ajustar algoritmos definidos de aprendizaje de la red neuronal con datos almacenados.		
Actividades	**Herramientas**	**Resultados**
Entrenamiento de la Red Neuronal Convolucional	Software Python, Banco de Datos	Red Neuronal Funcional para la clasificación de Mancha Foliar en plantas de maíz.

Objetivo 7: Validar la efectividad del modelo de aprendizaje VGG16 implementando muestras nuevas de maíz no consideradas al principio de la investigación.		
Actividades	**Herramientas**	**Resultados**
Calificación de muestras nuevas de maíz real.	Agricultores dedicados al cultivo de maíz.	Calificación de la calidad del maíz bajo las muestras de estudio, según el método seleccionado.
Realizar validación	Software Python, Agricultores	Descripción de las medidas de desempeño obtenidas
Validación de resultados y conclusiones	Microsoft Word	Reporte Final, Imágenes.

Capitulo 6

Resultados

- Resultados

6. RESULTADO

Los resultados del desarrollo, diseño e implementación de una Red Neuronal para la clasificación y predicción de imágenes para la predicción de enfermedades de las plantas del maíz, fueron la conformación de 2 programas:

El primero corresponde al Pre procesamiento de imágenes, en el cual, se hace un procesado para la eliminación de ruido, mejora de resolución y recorte de imágenes, este desarrollo permite presentar un tratamiento a las imágenes antes de ser analizadas por la red neuronal, en la cual, sigue un tratamiento de conversión a escala de grises, para luego ser analizadas por el Filtro Anisotrópico, una vez realizada la filtración , se pasa por un tratamiento de bordes de Sobel, el cual nos permitirá enmarcar las regiones de las imágenes analizadas para por último, realizar un corte a la zona más sobresaliente y guardar este último dentro de una nueva carpeta, la cual, se agregará a la carpeta final de las imágenes de prueba para el análisis de la red neuronal.

El segundo programa, se enfoca principalmente en el desarrollo de la red neuronal a través del modelo VGG16, el cual, nos permitirá analizar y predecir a partir de las imágenes, si la planta de maíz está enferma o no, su principal función es analizar una carpeta de imágenes y a través de esta, crear y guardar lo que se conoce como CodeBooks, los cuales, son archivos que guardan toda la información matemática de las imágenes , las cuales nos permitirá realizar las predicciones de la red neuronal.

Tomando en cuenta la tecnología utilizada y los resultados obtenidos con esta, es posible también establecer los siguientes resultados:

1.- La implementación de redes neuronales a la agricultura favorece a un mejor análisis, accesibilidad y manejo de datos a partir de imágenes.

2.- La información es contenida en Codebooks, la cual es rápida de decodificar para poder acceder a los datos contenidos dentro de este.

3.- Los codebooks son reutilizables, esto implica, al agregar más datos sobre otras enfermedades, estos no pierden la información actual, si no, la agrega a los conocimientos ya establecidos dentro de la red neuronal.

4.- La aplicación de esta tecnología es de bajo costo, ya que, al ser de uso libre, no implica ningún costo su uso para distintas áreas de aplicación.

5.- El diseño de implementación de estos puede variar, con el avance de las nuevas tecnologías este ha crecido en cuanto a los diseños de implementación, ya que, al ser adaptable, cada usuario le da un enfoque característico. El uso de redes neuronales al ser una tecnología en constante cambio dentro de sus algoritmos laborales, tiene algunas desventajas a tomar en cuenta:

1.- Es poco práctico el desarrollo e implementación de una red neuronal con varias capas, debido a que, para el desarrollo e implementación de estos, se requiere de un equipo de cómputo con ciertas características para no tener ningún problema en la etapa de pruebas e implementación.

2.- Actualmente es poco utilizado dentro de la agricultura, por ello, el desarrollo de esta tecnología no ha sido bien implementada aun, provocando así, que no existan estándares de creación y uso.

3.- La falta de banco de imágenes digitales hace que el desarrollo de estas redes neuronales sea poco eficaz al momento de su implementación, ya que depende de mucha información para poder realizar predicciones fiables al usuario final.

A continuación, se presentan las imágenes resultantes de cada uno de los pasos realizados para llevar a cabo, desde el Pre procesamiento de Imágenes hasta el entrenamiento de la Red Neuronal:

1. Realizar trabajo de campo, la cual consiste en obtener imágenes digitales de las plantas de maíz físicas a través de la cámara integrada en un Smartphone Samsung S8.

a)

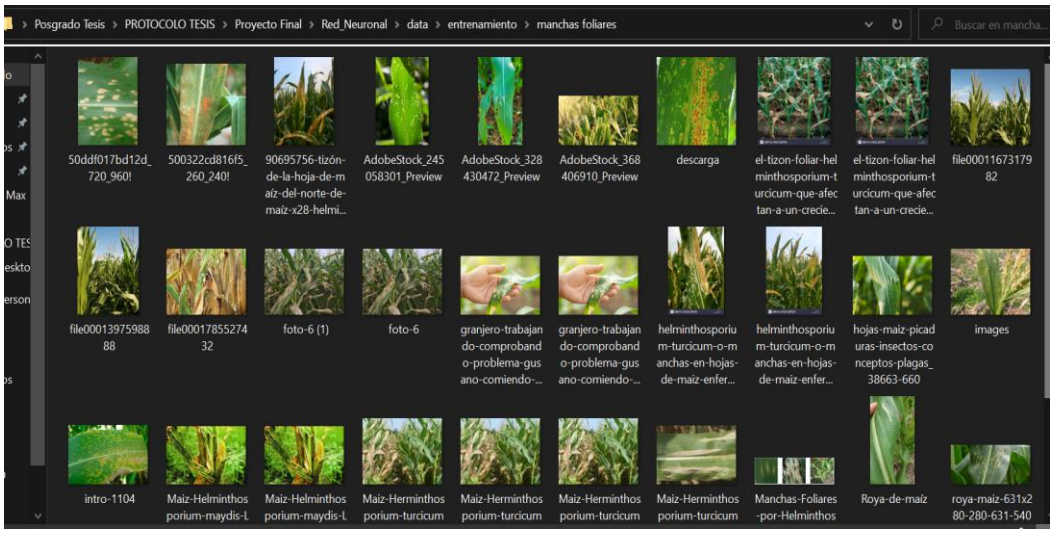
b)

c)

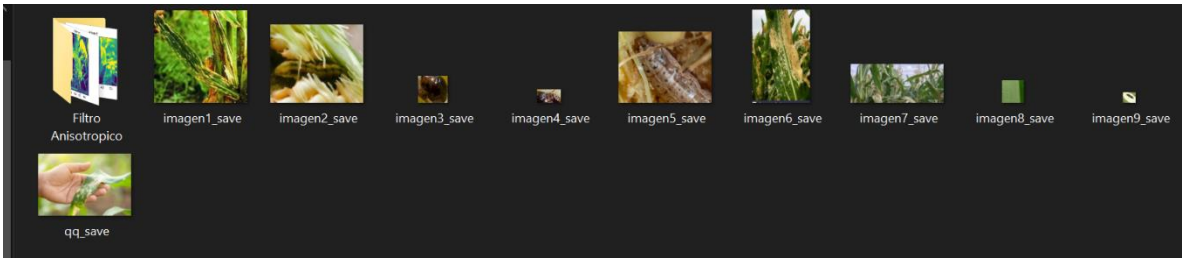
Fig. XXIX. Imágenes digitales resultantes de la cámara fotográfica del Samsung s8:
a) Diseño del Smartphone Samsung s8, b) Toma panorámica de las plantas de maíz, c) Vista de las hojas de una planta de maíz.

2. Una vez realizada la toma de muestras, estas se almacenan dentro del disco duro del equipo de cómputo, para luego escoger una imagen, la que el usuario considere viable para guardar en el nuevo banco de datos para la red neuronal, dicha imagen se carga dentro del programa para poder ser pre procesada antes de analizar con la red neuronal.

a)



b)



c)

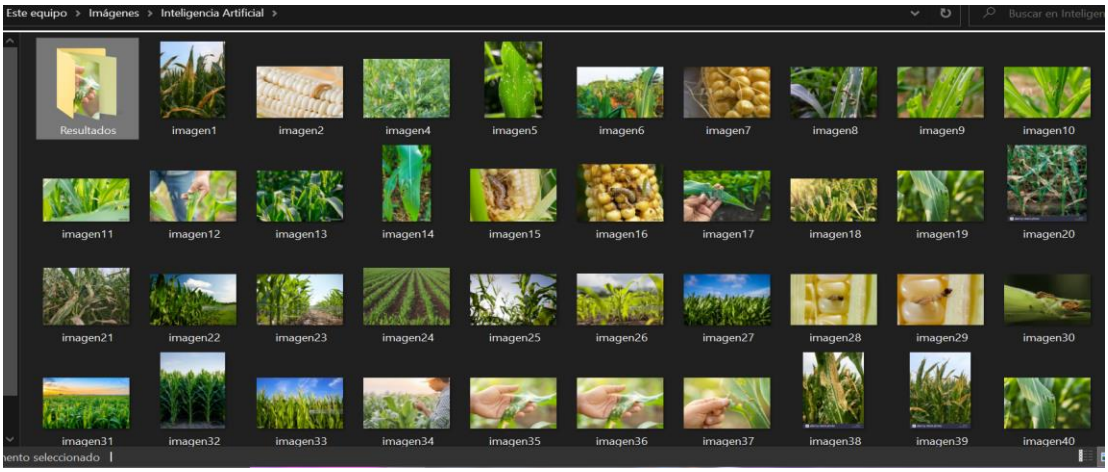


Fig. XXI. Rutas de Acceso para las Imágenes a Analizar para el Pre-Procesamiento: a) Muestras para Analizar, b) Ruta de acceso a Resultados, c) Vista de las imágenes nuevas de Resultados Finales.

3. Para comenzar con el análisis del Pre Procesamiento, primero se aplica un escalamiento a gris a la imagen original para poder pasar a la siguiente fase del análisis del código:

a)

```
import cv2 as cv
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
from tqdm import tqdm

#cargamos la imagen a utilizar
def get_image(path):
    img=cv.imread(path)
    #gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)

    return img
#procesamiento en escala de grises
def rgb2gray(img):
    h=img.shape[0]
    w=img.shape[1]
    img1=np.zeros((h,w),np.uint8)
    for i in range(h):
        for j in range(w):
            img1[i,j]=0.144*img[i,j,0]+0.587*img[i,j,1]+0.299*img[1,j,1]
    return img1

#calcular el nucleo de convolucion gaussiano
def gausskernel(size):
    sigma=1.0
    gausskernel=np.zeros((size,size),np.float32)
    for i in range(size):
        for j in range(size):
            norm=math.pow(i-1,2)+pow(j-1,2)
            gausskernel[i,j]=math.exp(-norm/(2*math.pow(sigma,2)))#Encuentra la convolu
            sum=np.sum(gausskernel)#sum
```

b)

Fig. XXII. Códigos de Pre Procesamiento de Imágenes:
a) Lectura de la imagen a leer; Conversión a Escala de Gris, b) Resultado de conversión de imagen original a escala de Grises.

4. Luego de la conversión a escala de grises de la imagen, se aplica un filtro gaussiano a la imagen para poder eliminar el ruido de la imagen de ser requerido.

a)

```
#calcular el nucleo de convolucion gaussiano
def gausskernel(size):
    sigma=1.0
    gausskernel=np.zeros((size,size),np.float32)
    for i in range(size):
        for j in range(size):
            norm=math.pow(i-1,2)+pow(j-1,2)
            gausskernel[i,j]=math.exp(-norm/(2*math.pow(sigma,2)))#Encuentra la convolucion
            sum=np.sum(gausskernel)#sum
            kernel=gausskernel/sum #normalizacion
    return kernel

#filtro gaussiano
def gauss(img):
    h=img.shape[0]
    w=img.shape[1]
    blurred=np.zeros((h,w),np.uint8)
    kernel=gausskernel(9)#calcular kernel de convolucion gaussiana
    for i in range(1,h-1):
        for j in range(1,w-1):
            sum=0
            for k in range(-1,2):
                for l in range(-1,2):
                    sum+=img[i+k,j+1]*kernel[k+1,l+1] #filtro gaussiano
                    blurred[i,j]=sum
    return blurred
```

b)

Fig. XXIII. Código de Pre Procesamiento de Imágenes:
a) **Código de aplicación del Filtro Gaussiano a la imagen en escala de gris, b) Resultado del Filtro Gaussiano.**

5. Debido a que el Filtro Gaussiano no causaba gran cambio a las imágenes resultantes, se optó por aplicar un filtro anisotrópico a las imágenes, para la eliminación de ruido en caso de tener, pero así también, aplicar un mejoramiento a la resolución de la imagen final.

a)

```
    # ...you could always diffuse each color channel independently if you
    # really want
    if img.ndim == 3:
        warnings.warn("Only grayscale images allowed, converting to 2D matrix")
        img = img.mean(2)

    # initialize output array
    img = img.astype('float32')
    imgout = img.copy()

    # initialize some internal variables
    deltaS = np.zeros_like(imgout)
    deltaE = deltaS.copy()
    NS = deltaS.copy()
    EW = deltaS.copy()
    gS = np.ones_like(imgout)
    gE = gS.copy()

    # create the plot figure, if requested
    if ploton:
        import pylab as pl
        from time import sleep

        fig = pl.figure(figsize=(20,5.5),num="Anisotropic diffusion")
        ax1,ax2 = fig.add_subplot(1,2,1),fig.add_subplot(1,2,2)

        ax1.imshow(img,interpolation='nearest')
        ih = ax2.imshow(imgout,interpolation='nearest',animated=True)
        ax1.set_title("Original image")
        ax2.set_title("Iteration 0")

        fig.canvas.draw()
```

b)

```
265         # update matrices
266         D = gD*deltaD
267         E = gE*deltaE
268         S = gS*deltaS
269
270         # subtract a copy that has been shifted 'Up/North/West' by one
271         # pixel. don't as questions. just do it. trust me.
272         UD[:] = D
273         NS[:] = S
274         EW[:] = E
275         UD[1:,: ,: ] -= D[:-1,:  ,:  ]
276         NS[: ,1:,: ] -= S[:  ,:-1,:  ]
277         EW[: ,: ,1:] -= E[:  ,:  ,:-1]
278
279         # update the image
280         stackout += gamma*(UD+NS+EW)
281
282         if ploton:
283             iterstring = "Iteration %i" %(ii+1)
284             ih.set_data(stackout[showplane,...].squeeze())
285             ax2.set_title(iterstring)
286             fig.canvas.draw()
287             # sleep(0.01)
288
289     return stackout
290
291 img=io.imread('C:/Users/andre/Pictures/Inteligencia Artificial/imagen50.jpg')
292 img=img.astype(float)
293 #img=img[300:600,300:600]
294 m=np.mean(img)
295 s=np.std(img)
296 nimg=(img-m)/s
```

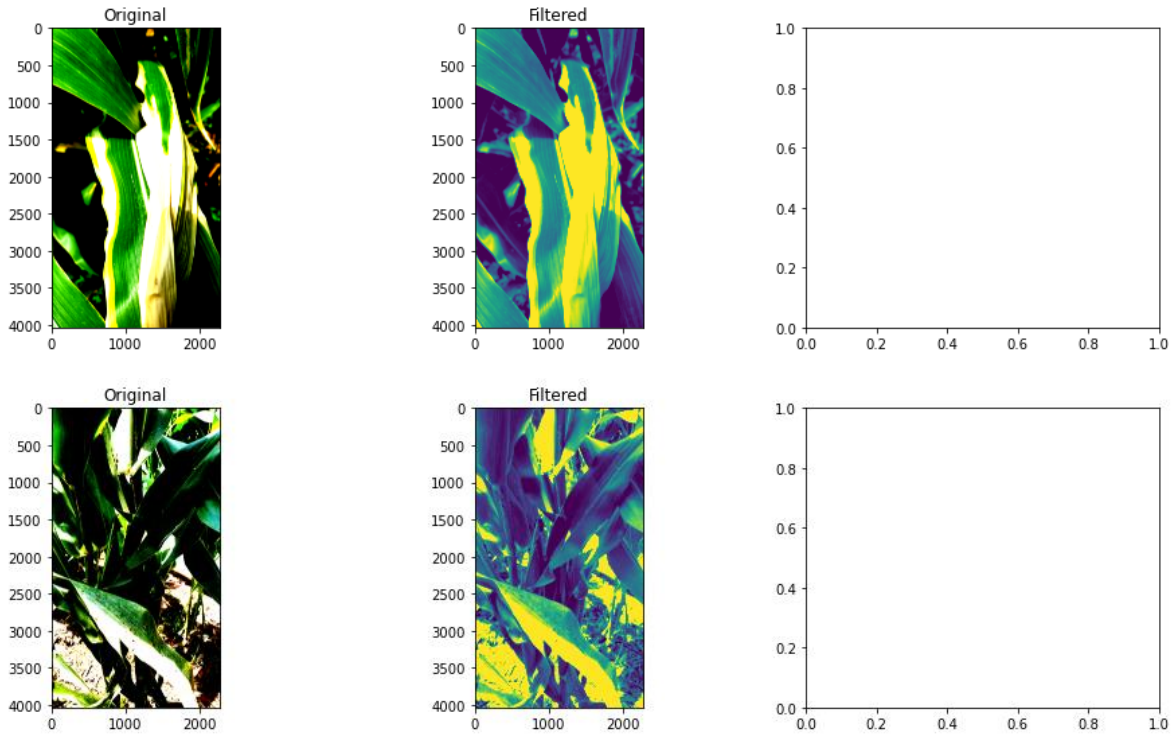


Fig. XXIV. Código de Filtro Anisotrópico:

a) Código de aplicación del Filtro Anisotrópico a la imagen en escala de gris, b) Resultado del Filtro Anisotrópico.

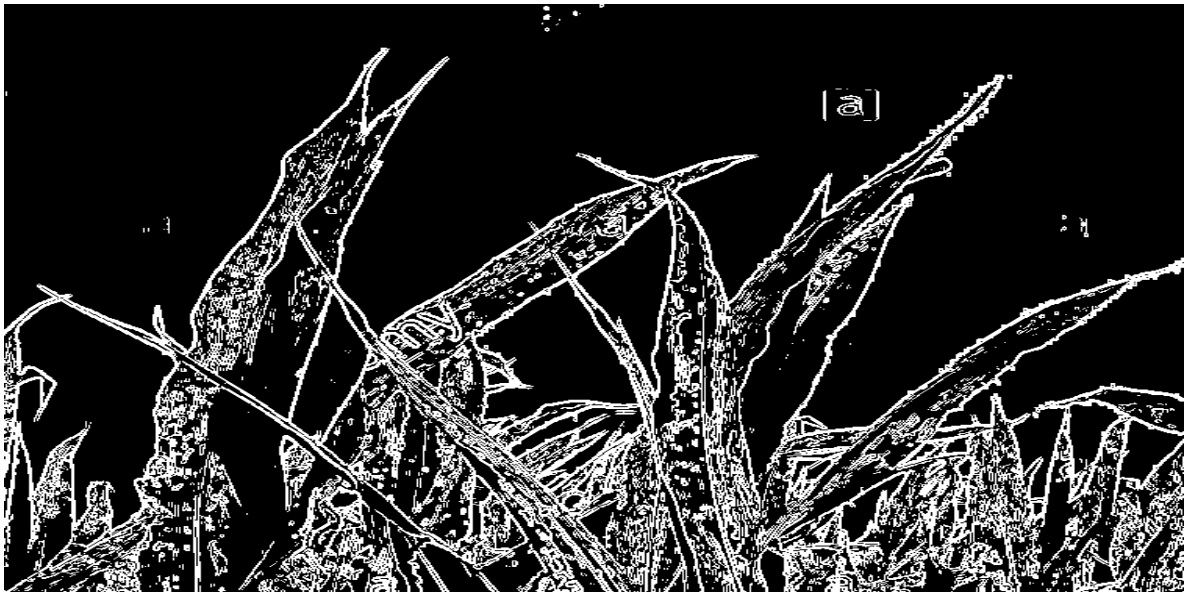
6. Una vez aplicado los filtros para la eliminación de ruido, se procede a realizar la detección de bordes dentro de la imagen a través del algoritmo de bordes de sobel tanto: vertical, horizontal, una vez obtenidos los bordes verticales y horizontales se hace una última detección combinando los bordes Verticales y Horizontales.

a)

```
    return blurred
#una vez hecho el filtro para eliminacion de ruidos y mejoramiento de la imagen
#procedemos a realizar la deteccion de filtros de sobel
#primero definimos en vertical
def sobel_v(blurred,threshold):
    G_x=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    rows=np.size(blurred,0)
    col=np.size(blurred,1)
    mag=np.zeros(blurred.shape)
    for i in range(0,rows-2):
        for j in range(0,col-2):
            v=sum(sum(G_x*blurred[i:i+3,j:j+3])) #vertical
            mag[i+1,j+1]=v

    for p in range(0,rows):
        for q in range(0,col):
            if mag[p,q]<threshold:
                mag[p,q]=0
    return mag
```

```
    return blurred
#una vez hecho el filtro para eliminacion de ruidos y mejoramiento de la imagen
#procedemos a realizar la deteccion de filtros de sobel
#primero definimos en vertical
def sobel_v(blurred,threshold):
    G_x=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    rows=np.size(blurred,0)
    col=np.size(blurred,1)
    mag=np.zeros(blurred.shape)
    for i in range(0,rows-2):
        for j in range(0,col-2):
            v=sum(sum(G_x*blurred[i:i+3,j:j+3])) #vertical
            mag[i+1,j+1]=v

    for p in range(0,rows):
        for q in range(0,col):
            if mag[p,q]<threshold:
                mag[p,q]=0
    return mag
#deteccion de bordes en horizontal
def sobel_h(blurred,threshold):
    G_y=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    rows=np.size(blurred,0)
    col=np.size(blurred,1)
    mag=np.zeros(blurred.shape)
    for i in range(0,rows-2):
        for j in range(0,col-2):
            h=sum(sum(G_y*blurred[i:i+3,j:j+3])) #horizontal
            mag[i+1,j+1]=h

    for p in range(0,rows):
        for q in range(0,col):
            if mag[p,q]<threshold:
                mag[p,q]=0
```

b)



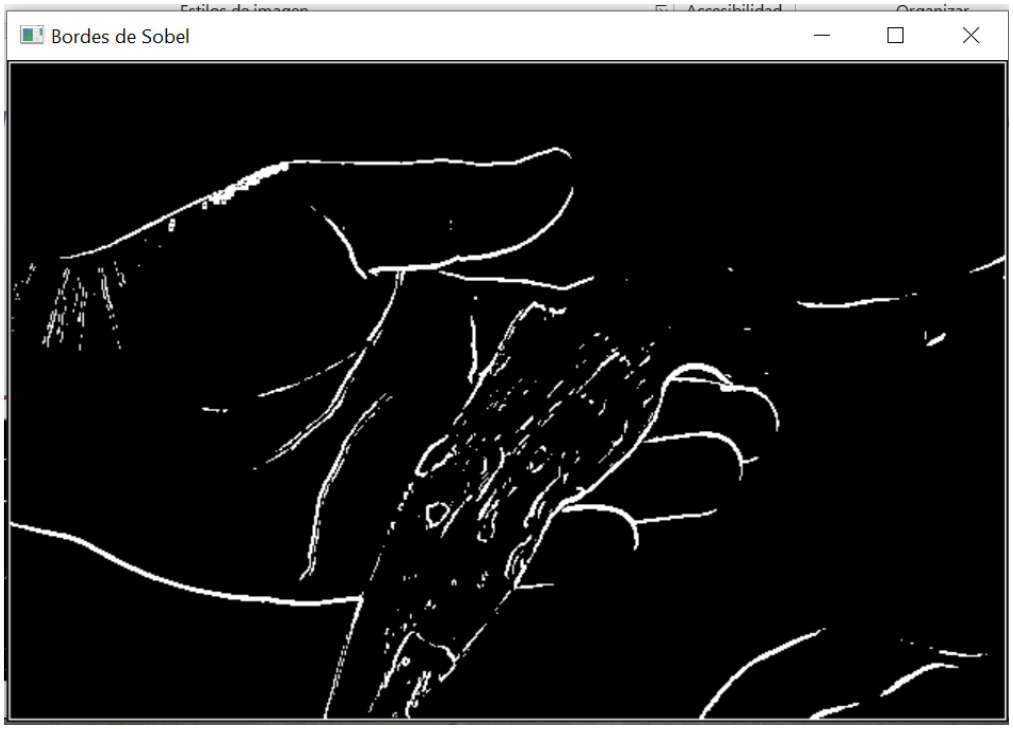


Fig. XXV. Código de Bordes de Sobel:

a) Código de aplicación de los bordes de Sobel: Vertical, Horizontal y ambos sentidos a la imagen filtrada, b) Resultados de los Bordes de Sobel

7. Luego de realizar el mejoramiento a la imagen, se procede a realizar una expansión de la imagen a través de un algoritmo que se detecta a partir de los bordes (binarización), a encontrar una pérdida de detalles por ello se expande para poder encontrar la región y marcarla.

a)

```
#obtenemos una binarizacion de la imagen
#a traves de un filtro paso bajo
#aqui seleccionamos el nucleo ELLIPSE y se usa la operacion

def Thresh_and_blur(mag): #blurred,
    h=mag.shape[0]
    w=mag.shape[1]

    #Determinamos unicamente el umbral
    a=90
    new_image=np.zeros((h,w),np.uint8)
    filtro=cv.GaussianBlur(mag,(9,9),0)
    for i in tqdm(range(h)):
        for j in range(w):
            if(filtro[i,j]>a):
                new_image[i,j]=255
            else:
                new_image[i,j]=0
    return new_image
```

b)

Fig. XXVI. Código de Binarización:
a) Código de aplicación de binarización a los bordes de sobel, b) Resultados de la imagen binarizada.

8. Una vez detectada la región, se procede a encontrar el contorno del área y dibujar una caja alrededor de la imagen cuando detecta los bordes.

a)

```
#en este apartado, encontramos el contorno del area
def findcnts_and_box_point(closed):
    (cnts,_)=cv.findContours(closed.copy(), cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
    c=sorted(cnts,key=cv.contourArea, reverse=True)[0]
    #computariza la rotacion del la caja a lo largo del contorno
    rect=cv.minAreaRect(c)
    box=np.int0(cv.boxPoints(rect))

    return box
def drawcnts_and_cut(original_img,box):
    #dibuja una caja alrededor de la imagen cuando detecte los bordes
    draw_img=cv.drawContours(original_img.copy(),[box],-1,(0,0,255),3)

    Xs=[i[0] for i in box]
    Ys=[i[1] for i in box]
    x1=min(Xs)
    x2=max(Xs)
    y1=min(Ys)
    y2=max(Ys)
    hight=y2-y1
    width=x2-x1
    crop_img=original_img[y1:y1+hight, x1:x1+width]

    return draw_img,crop_img
```

b)

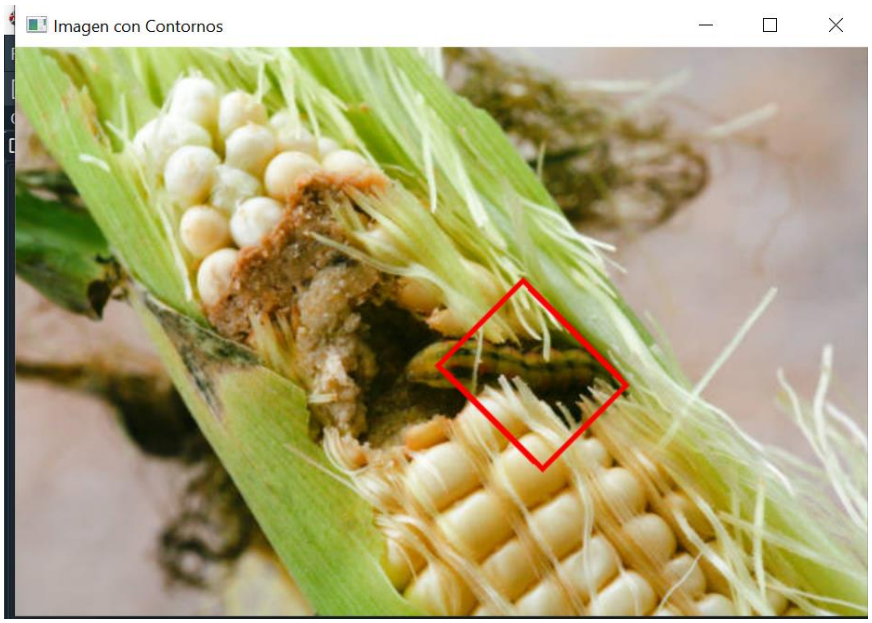
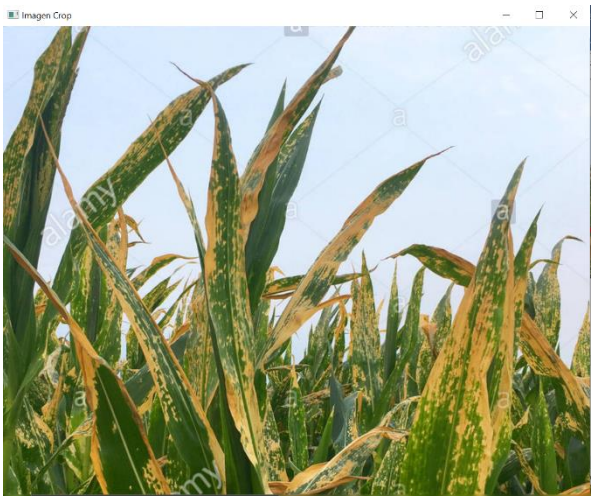


Fig. XXVII. Código para marcar Contorno del Área seleccionada:
a) Código de aplicación de contornos a una zona en específico, b) Resultados de la aplicación del contorno al área señalada.

9. Al final realiza un corte de la región encontrada anteriormente y pasa a guardarse en una nueva carpeta de resultados, la cual, pasara a formar parte de las imágenes a analizar de la Red Neuronal.

a)



b)

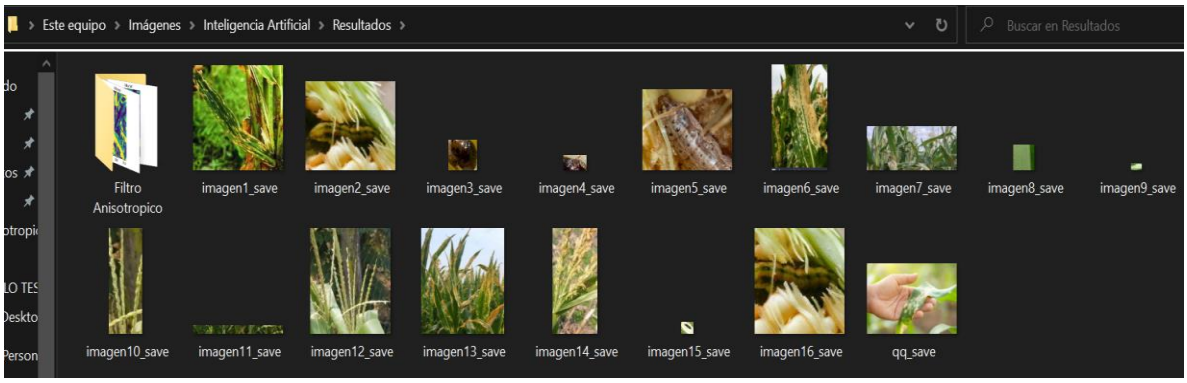


Fig. XXVIII. Corte de la zona marcada:

a) Resultado de los cortes finales realizados a la zona marcada, b) Ruta de acceso donde se podrá acceder a las nuevas imágenes basadas en los cortes realizados a la imagen original.

10. Una vez hecho el proceso de Pre Procesamiento a todas las imágenes, se procede a trabajar con la red neuronal, a la cual se le asignan dos carpetas, una de entrenamiento y otra de validación.

a)

```
data_entrenamiento='./data/entrenamiento'
data_validacion='./data/validacion'

#Parametros
epocas=8
longitud,altura=100,100
batch_size=32
pasos=100 #numero de veces que procesa en cada epoca
pasos_validacion=300
filtrosConv1=32
filtrosConv2=64
tamano_filtro1=(3,3)
tamano_filtro2=(2,2)
tamano_pool=(2,2)
clases=3 #carpetas a analizar
lr=0.0004 #ajustes para acercarse a solucion optima

#pre procesamiento de imagenes
```

Fig. XXIX. Declaración de la ruta de acceso al banco de datos:

a) Se declara la ruta de acceso al banco de datos para el entrenamiento de la Red Neuronal, así como los parámetros para su entrenamiento.

11. En dichas carpetas, estarán contenidas todas las imágenes a utilizar para poder realizar el entrenamiento de la red neuronal, luego, se realiza la declaración de variables para poder activar el modelo de entrenamiento VGG16.

a)

```
                                                batch_size=32, class_mode='categorical')
#creamos red convolucional (modelo preentrenado de google)
vgg=applications.vgg16.VGG16()
#vgg=applications.vgg16.VGG16()# Ejemplo del modelo
model=Sequential()
#cargamos las capas del modelo preentrenado del modelo secuencial  y se quita la ultima capa
for capa in vgg.layers:
    model.add(capa)
model.layers.pop()
#solo se quiere que entrene la capa que se esta añadiendo
for layer in model.layers:
    layer.trainable=False

#añadimos nuestra propia capa
model.add(Dense(256,activation='softmax'))

#Compilamos el modelo
model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(lr=0.0004),
              metrics=['accuracy'])
```

b)

```
#crear la red CNN
model=Sequential()

model.add(Convolution2D(filtrosConv1, tamano_filtro1, padding='same', input_shape=(altura,longitud,3),

model.add(MaxPooling2D(pool_size=tamano_pool))

model.add(Convolution2D(filtrosConv2, tamano_filtro2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=tamano_pool))

model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(clases, activation='softmax'))

model.compile(loss='categorical_crossentropy',
          optimizer=optimizers.Adam(lr=lr),
          metrics=['accuracy'])

model.fit(imagen_entrenamiento,
        steps_per_epoch=pasos,
        epochs=epocs,
        validation_data=imagen_validacion,
        validation_steps=pasos_validacion)


dir='./modelo/'
```

Fig. XXX. Activación del modelo VGG16:
a) Parámetros para la activación de la red neuronal, b) Parámetros para la lectura de las variables de entrenamiento.

12. Una vez analizadas todas las imágenes contenidas en las carpetas ya mencionadas, esta procederá a guardar dos archivos con extensión ".h5", una que contendrá el modelo de la red neuronal entrenada y otra, que contendrá los datos matemáticos de cada una de las imágenes con las que se entrenó, denominada "pesos". Con estos archivos se procederá a realizar la predicción con la inteligencia artificial.

a)

```
In [26]: runfile('C:/Users/Helen/Documents/Python Scripts/Red Neuronal/entrenar.py', wdir='C:/
Users/Helen/Documents/Python Scripts/Red Neuronal')
Found 75 images belonging to 4 classes.
Found 75 images belonging to 4 classes.
Epoch 1/20
WARNING:tensorflow:5 out of the last 13 calls to <function
Model.make_train_function.<locals>.train_function at 0x000001EA3677BDC0> triggered tf.function
retracing. Tracing is expensive and the excessive number of tracings could be due to (1)
creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3)
passing Python objects instead of tensors. For (1), please define your @tf.function outside of
the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to https://
www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args and https://
www.tensorflow.org/api_docs/python/tf/function for  more details.
   3/1000 [..............................] - ETA: 31:57 - loss: 1.8439 - accuracy:
0.3467WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case,
20000 batches). You may need to use the repeat() function when building your dataset.
WARNING:tensorflow:5 out of the last 13 calls to <function
Model.make_test_function.<locals>.test_function at 0x000001EA29357B80> triggered tf.function
retracing. Tracing is expensive and the excessive number of tracings could be due to (1)
.3467WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
lataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case,
0000 batches). You may need to use the repeat() function when building your dataset.
ARNING:tensorflow:5 out of the last 13 calls to <function
lodel.make_test_function.<locals>.test_function at 0x000001EA29357B80> triggered tf.function
etracing. Tracing is expensive and the excessive number of tracings could be due to (1)
reating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3)
assing Python objects instead of tensors. For (1), please define your @tf.function outside of
he loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument
hapes that can avoid unnecessary retracing. For (3), please refer to https://
ww.tensorflow.org/tutorials/customization/performance#python_or_tensor_args and https://
ww.tensorflow.org/api_docs/python/tf/function for  more details.
ARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
lataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 300
atches). You may need to use the repeat() function when building your dataset.
  3/1000 [..............................] - 20s 7s/step - loss: 1.8439 - accuracy: 0.3467 -
al_loss: 2.5099 - val_accuracy: 0.4133
```

b)

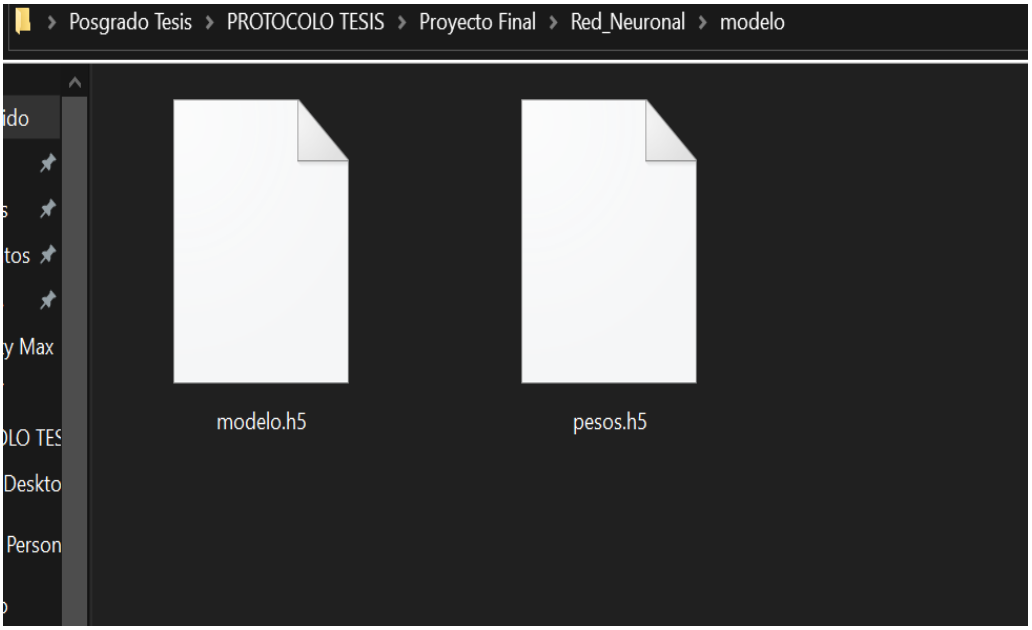


Fig. XXXI. Resultado del entrenamiento:
a) Creación de archivos ".H5" para su lectura y posterior entrenamiento, b) código de lectura de los archivos .H5, los cuales sirven para la predicción de la rede neuronal.

13. Para poder realizar la predicción de las imágenes, a la red neuronal tendrá acceso al archivo con los "pesos" de las imágenes, al momento de darle una nueva imagen, lo que hará, será una comparación de la nueva imagen con los "pesos" y a partir de ello, dar una predicción final la cual dirá si la planta de maíz contiene plaga o esta sana.

a)

```
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.python.keras.models import Sequential #libreria para redes neuronales secuenciales
from tensorflow.keras.models import load_model
from tensorflow.python.keras import applications
from tensorflow.python.keras.layers import Dense

longitud, altura=224, 224
weights_model='./modelo/pesos.h5'
vgg=applications.vgg16.VGG16()
cnn=Sequential()
for capa in vgg.layers:
    cnn.add(capa)
cnn.layers.pop()
for layer in cnn.layers:
    layer.trainable=False
cnn.add(Dense(2,activation='softmax'))

cnn.load_weights(weights_model)

def predict(file):
    x=load_img(file,target_size=(longitud,altura))
    x=img_to_array(x)
    x=np.expand_dims(x,axis=0)
    array=cnn.predict(x)
    result=array[0]
    answer=np.argmax(result)

    if answer==0:
        print("Mancha Foliar")
    elif answer==1:
        print("Gusano Trozador")
```

b)

```
[[ 84, 216, 209],
 [ 85, 217, 207],
 [ 70, 203, 193],
 ...,
 [ 28,  52,  40],
 [ 28,  53,  43],
 [ 27,  52,  42]]], dtype=uint8)]
```


```
#Prediccion
predictions_single= model.predict(img)
print(predictions_single)
print(np.sum(predictions_single))
print(np.argmax(predictions_single))
print(class_names[np.argmax(predictions_single)])
```

```
WARNING:tensorflow:Model was constructed with shape (None, 150, 150, 3) for input KerasTensor(type_spec=TensorSpec(shape=(N
[[0. 1.]]
1.0
1
Manchas Foliares
```

Fig. XXXII. Resultado de Predicción:
a) Código para la lectura de los archivos .H5 y realizar la predicción, b) Predicción realizada con los datos ya cargados a la red neuronal.

Capitulo 7

Conclusiones

- Conclusión final de la investigación y trabajos a futuro.

7. CONCLUSIÓN

Este proyecto surgió debido a la necesidad de contar con un Sistema de Detección y Clasificación de Imágenes que ayuden al avance tecnológico utilizado en la Agricultura de Precisión de México, para así poder formar parte importante en el cuidado de los cultivos de maíz de riego que son parte importante de la economía del país, por ello, cultivar productos de calidad es parte fundamental de sus producto final, para ello se diseñó una Red Neuronal formado por 2 fases: El Pre- Procesamiento de imágenes y la Red Neuronal, conformando así un sistema complejo inteligente para la clasificación de imágenes.

A medida que el proyecto evolucionaba, varios algoritmos, componentes de la red neuronal y rutinas de programación fueron cambiando de manera acorde a las nuevas necesidades que surgían. La contribución más importante de este proyecto fue el desarrollo e implementación del método practico y novedoso del manejo de una Red Neuronal para la clasificación y almacenamiento de datos contenidos en las imágenes. Por lo anterior, se puede establecer de manera puntual las siguientes conclusiones:

- Se tiene una Red Neuronal adaptado al Pre Procesamiento de imágenes para obtener mejores resultados.
- Se desarrollo la programación para la utilización de métodos matemáticos de procesamiento de imágenes.
- Se llevaron a cabo las pruebas de funcionamiento del Sistema de Clasificación de imágenes a través de una Red Neuronal.
- Se dio soporte a la base de datos de imágenes para ser compartida entre la Red Neuronal y el Pre Procesamiento de Imágenes.
- Seguir desarrollando la adaptación de la Red Neuronal a un aplicativo móvil para ser utilizado en cualquier dispositivo móvil.

7.1 Trabajo a Futuro

El tiempo ha sido uno de los factores que más en cuenta se ha tenido para el desarrollo del trabajo, teniendo en cuenta la dificultad que entraña la identificación de imágenes basadas en su contenido, los resultados ofrecidos durante la investigación han sido aceptables ya que es capaz de identificar imágenes que se capturan en buenas condiciones.

El trabajo realizado, constituye una primera aproximación a la determinación de la calidad de las plantas de maíz mediante herramientas computacionales y establece un modelo para implementar en aplicaciones donde la calidad se determine mediante operaciones realizada por humanos.

Debido a que la investigación de campo se realizó durante la temporada de siembra de maíz tradicional, el tiempo establecido para poder realizar nuevas muestras era imposible, debido a la línea recta que sigue el proceso de cosecha tradicional. Sin embargo, se tienen muy en cuenta las ampliaciones que se pueden aplicar al sistema.

- **Integrar servicios cloud**. Actualmente diferentes plataformas ofrecen diferentes máquinas virtuales con todas las tecnologías necesarias para el aprendizaje profundo, tal es el caso de Google Colab, el cual es un producto que se puede usar para escribir y ejecutar códigos aleatorios de programas de Python usando un navegador web, es decir, una versión alojada en la nube de Cuaderno Jupyter. Ofrece la posibilidad de tener más acceso a diferentes bases de datos de imágenes, conectar al usuario a una máquina virtual desde su computador para la reducción considerable del consumo de recursos de la maquina y tiempos de entrenamiento.
- **Obtención de imágenes para un conjunto de datos**. En el módulo de creación de un conjunto de datos, se requiere añadir API´s ofrecidas por Google y Microsoft para la descarga de imágenes de distintas fuentes. Esta nueva funcionalidad facilita notablemente el trabajo de búsqueda de imágenes para el conjunto de datos.
- **Cargar una red convolucional**. Añadir una funcionalidad en el módulo de creación de una red convolucional, gracias al avance de los servicios cloud, es posible crear un módulo de carga de una red para que el usuario pueda cargar y descargar un modelo de red convolucional creado previamente con el fin de realizar modificaciones en este.
- **Uso de aplicaciones móviles**. Para la detección del estado de plaga que contiene la planta del maíz, a través del modelo entrenado, se puede obtener los pesos del modelo

para la predicción de las variables a las que fue entrenado. Esto permite, la creación de aplicaciones, que, desde un smartphone, permita a los consumidores, saber el estado de la planta, una herramienta importante para cosechadores no expertos o con un plan de cosecha tradicional.

- **Uso de servicios web**. Para poder crear un enlace directo a los servicios cloud y la aplicación móvil, es necesario, crear a través de lenguajes como React, JavaScript, PHP, Python, con la cual, se creará un aplicativo web hibrido que funcione como un aplicativo móvil, la cual, acceda a la base de datos en la nube de la red neuronal y pueda realizar predicciones en tiempo real.
- **Base de datos interna**. Debido a la ubicación de distintas parcelas y/o sistemas de riego, el acceso a internet es poco probable o nulo, por ello, se busca diseñar un algoritmo dentro del aplicativo móvil, el cual permita crear una base de datos interna, para el funcionamiento eficiente de la rede neuronal con o sin acceso a internet.
- **Aumento en la detección de plagas**. Agregar más módulos de plaga y/o enfermedades para los cultivos de plantas de maíz como: Gusano Trozador, Polillas, pudrición de raíces, pudrición de mazorca, etc.

Estos son los trabajos que se han planteado para una futura implementación. Sin embargo, con un estudio más profundo de las necesidades de un usuario avanzado, este número de propuestas de ampliarían. Hay que recordar también que las tecnologías como Keras, Tensor Flow y servicios Cloud están siempre en constante actualización, lo cual es muy posible que aparecerán nuevas propuestas de implementación.

Bibliografía

- Listado de cada una de las fuentes consultadas durante toda la investigación.

BIBLIOGRAFIA

- [1] Berrío M. Viviana., Mosquera T. Jemay., Alzate V. Diego F., 'Uso de drones para el analisis de imagines multiespectrales en agricultura de precision', ISSN 1692-7125. Volumen 13, No. 1, p.28-40, 2015, *Facultad de Ingenierías y Arquitectura, Universidad de Pamplona.*
- [2]INCyTU, 'Agricultura de Precisión', número 015, abril 2018
- [3] Chartuni M. Evandro, Magdalena Carlos, Manual de agricultura de precisión / IICA, PROCISUR, 2014.
- [4]Oliva R. Alejandro, Desarrollo de una aplicación de reconocimiento en imágenes utilizando Deep Learning con Open CV, ETSINF, 2017-2018
- [5] Escobar A. Emmanuel, Predicción de Agentes Patógenos en Plantas Ornamentales Utilizando Redes Neuronales, Tesis, División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Colima, Villa de Álvarez, Colima, Julio 2018.
- [6] Martínez Castillo Roger, Agricultura Tradicional Campesina: Características Ecológicas, Tecnología en Marcha, Vol. 21, N°3, Julio-septiembre 2008, P. 3-13.
- [7] Jain, A., *Fundamentals of Digital Image Processing*, Pretince-Hall, 1986.
- [8] González, R., Woods, P., *Digital Image Processing*, Pearson Prentice Hall, 2008.
- [9] Gabriel T. Juan, Constanza R. Liliana, Gerardo R. Diego, Restauración Digital de Imágenes mediante Ecuaciones Diferenciales Parciales, 2013.
- [10] Ortega M. Jhon F., Rivera V. Carlos E., Aplicación de un Filtro Adaptativo para minimizer el Ruido en una Imagen.
- [11] Peguero N. Pedro D., Realce y Restauración de Imagen, Extraído del PFC *Diseño y desarrollo de una aplicación de realce y restauración de imagen para la plataforma Android*.
- [12] Muhammad D. Giondal, YasirN. Khan., Early Pest Detection from Crop Using Image Processing and Computational Intelligence, FAST-UN, Junio 2015.
- [13]José M. Coleto M., Agricultura Convencional y Agriculturas Alternativas, 2004, P. 215-225.
- [14] Enfermedades del maíz: una guía para su identificación en el campo, 4ta Edición, Programa de Maíz del CIMMYT,2004.

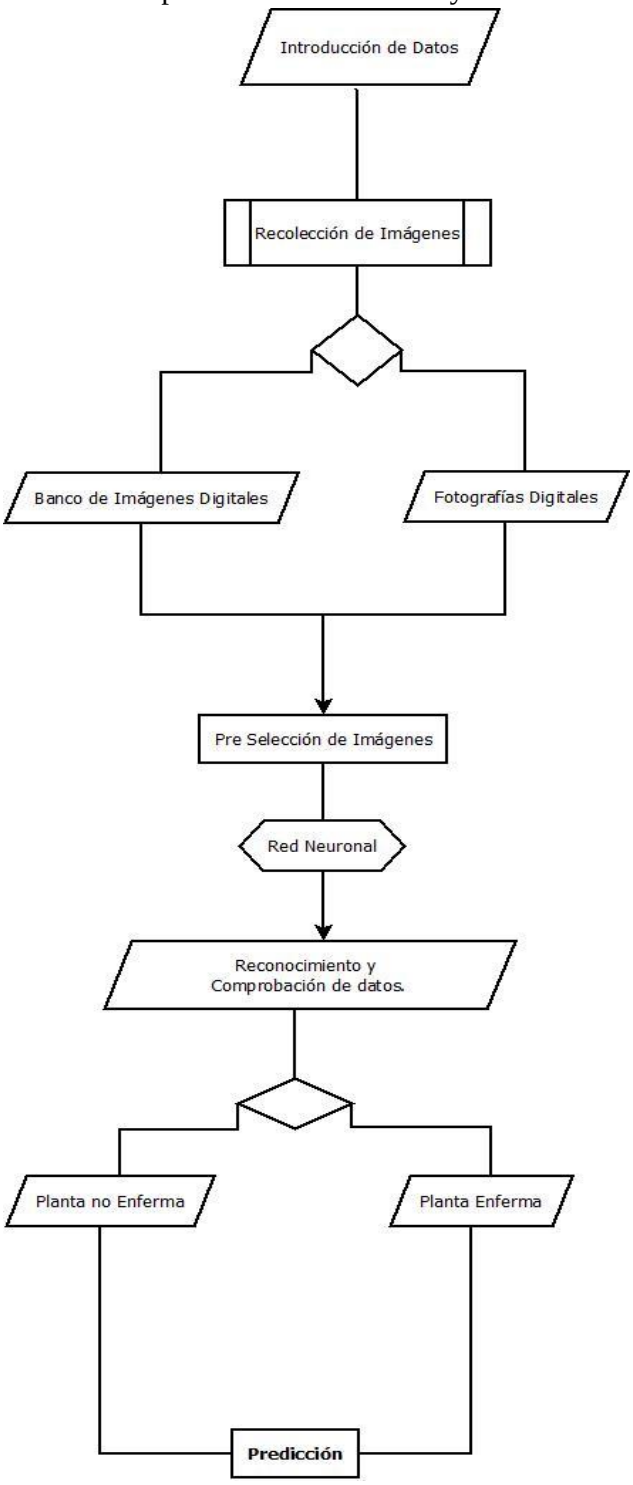
- [15] Diego A. Campanini G., Detección de objetos usando Redes Neuronales convolucionales junto con Random Forest y Support Vector Machines, Tesis, Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Ingeniería Eléctrica, 2018.
- [16] Pedro P. García G., Reconocimiento de Imágenes utilizando Redes Neuronales Artificiales, Tesis, Máster en Investigación en Informática, Facultad de Informática, Universidad Complutense de Madrid, 2012/2013.
- [17] Raybel H. Regalado, Clasificación de imágenes aéreas en la agricultura de precisión empleando redes neuronales, Trabajo de Diploma, Santa Clara, Universidad Central “Marta Abreu” de las Villas, junio 2018.
- [18] Pedro P. Cruz, Inteligencia Artificial con aplicaciones a la ingeniería, Primera Edición, Alfaomega Grupo Editor, S.A. de C.V., México, Julio 2010.
- [19] Humberto S. Azuela, Fernando R. Cortés, Inteligencia Artificial aplicada a Robótica y Automatización, Primera Edición, septiembre 2020.
- [20] Jean- Marie Mirebeau, Jérome Fenhrenbach, Laurent Risser, Shaza Tobji, Anisotropic Diffusion in ITK, Universidad Paris- Duaphine, Laboratorio Ceremade, CNRS,4 de marzo, 2015.
- [21] Joachim Weickert, Anisotropic Diffusion in Image Processing, Departamento de Ciencias de Computación, Universidad de Copenhagen, Dinamarca,1998.
- [22] Filtrado de Difusión Anisotrópica, 2 de octubre de 2007.
- [23] Tomi Sariola, Anisotropic difusión in image processing, 16 de septiembre, 2019.
- [24] Rodolfo Bongiovanni, Evandro C. Montovani, Stanley Best, Alvaro Roel, Agricultura de Precisión: Integrando conocimientos para una agricultura moderna y sustentable, Montevideo: PROCISUR/IICA, 2006.
- [25] José R. Mejía V., Apuntes de Procesamiento Digital de Imágenes (Primer borrador), Universidad Autónoma de San Luis Potosí, enero,2005.

Anexos

- Apartado donde se podrá consultar los diagramas resultantes de la investigación, así como los códigos empleados para la realización del sistema de reconocimiento de imágenes.

ANEXO I
DIAGRAMAS

Funcionamiento de una Red Neuronal para el reconocimiento y clasificación de imágenes.



ANEXO II: CODIGOS DE DESARROLLO

Código de Pre Procesamiento de Imágenes: En esta primera etapa, se ingresan las imágenes para realizar un tratamiento de mejoramiento en caso de haber ruido en las imágenes, realizar recortes para obtener nuevas muestras y una comparación de los filtros Gaussiano y Anisotrópico.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jun 19 19:32:46 2021

@author: Andrés González
"""


import cv2 as cv
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
from tqdm import tqdm

#cargamos la imagen a utilizar
def get_image(path):
    img=cv.imread(path)
    #gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)

    return img
#procesamiento en escala de grises
def rgb2gray(img):
    h=img.shape[0]
    w=img.shape[1]
    img1=np.zeros((h,w),np.uint8)
    for i in range(h):
        for j in range(w):
            img1[i,j]=0.144*img[i,j,0]+0.587*img[i,j,1]+0.299*img[1,j,1]
    return img1

#calcular el nucleo de convolucion gaussiano
def gausskernel(size):
    sigma=1.0
    gausskernel=np.zeros((size,size),np.float32)
    for i in range(size):
        for j in range(size):
            norm=math.pow(i-1,2)+pow(j-1,2)
            gausskernel[i,j]=math.exp(-norm/(2*math.pow(sigma,2)))#Encuentra la convolucion
            sum=np.sum(gausskernel)#sum
            kernel=gausskernel/sum #normalizacion
```

```
    return kernel

#filtro gaussiano
def gauss(img):
    h=img.shape[0]
    w=img.shape[1]
    blurred=np.zeros((h,w),np.uint8)
    kernel=gausskernel(9)#calcular kernel de convolucion gaussiana
    for i in range(1,h-1):
        for j in range(1,w-1):
            sum=0
            for k in range(-1,2):
                for l in range(-1,2):
                    sum+=img[i+k,j+1]*kernel[k+1,l+1] #filtro gaussiano
                    blurred[i,j]=sum
    return blurred
#una vez hecho el filtro para eliminacion de ruidos y mejoramiento de la imagen
#procedemos a realizar la deteccion de filtros de sobel
#primero definimos en vertical
def sobel_v(blurred,threshold):
    G_x=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    rows=np.size(blurred,0)
    col=np.size(blurred,1)
    mag=np.zeros(blurred.shape)
    for i in range(0,rows-2):
        for j in range(0,col-2):
            v=sum(sum(G_x*blurred[i:i+3,j:j+3])) #vertical
            mag[i+1,j+1]=v

    for p in range(0,rows):
        for q in range(0,col):
            if mag[p,q]<threshold:
                mag[p,q]=0
    return mag
#deteccion de bordes en horizontal
def sobel_h(blurred,threshold):
    G_y=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    rows=np.size(blurred,0)
    col=np.size(blurred,1)
    mag=np.zeros(blurred.shape)
    for i in range(0,rows-2):
        for j in range(0,col-2):
            h=sum(sum(G_y*blurred[i:i+3,j:j+3])) #horizontal
            mag[i+1,j+1]=h

    for p in range(0,rows):
        for q in range(0,col):
            if mag[p,q]<threshold:
                mag[p,q]=0
```

```
    return mag
#una vez detectado tanto en vertical y horizontal
#procedemos a combinar ambos metodos para obtener uno solo
def sobel(blurred,threshold):
    G_x=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    G_y=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    rows=np.size(blurred,0)
    col=np.size(blurred,1)
    mag=np.zeros(blurred.shape)
    for i in range(0,rows-2):
        for j in range(0,col-2):
            v=sum(sum(G_x*blurred[i:i+3,j:j+3])) #vertical
            h=sum(sum(G_y*blurred[i:i+3,j:j+3])) #horizontal
            mag[i+1,j+1]=np.sqrt((v**2)+(h**2))

    for p in range(0,rows):
        for q in range(0,col):
            if mag[p,q]<threshold:
                mag[p,q]=0
    return mag
#obtenemos una binarizacion de la imagen
#a traves de un filtro paso bajo
#aqui seleccionamos el nucleo ELLIPSE y se usa la operacion

def Thresh_and_blur(mag): #blurred,
    h=mag.shape[0]
    w=mag.shape[1]

    #Determinamos unicamente el umbral
    a=90
    new_image=np.zeros((h,w),np.uint8)
    filtro=cv.GaussianBlur(mag,(9,9),0)
    for i in tqdm(range(h)):
        for j in range(w):
            if(filtro[i,j]>a):
                new_image[i,j]=255
            else:
                new_image[i,j]=0
    return new_image

def image_morphology(thresh):
    #a partir de la imagen anterior, se puede encontrar que comparando
    #con la imagen original, encontramos que existe una perdida de detalles
    #por ello se debe expandir y realizar 4 veces la corrosion morfologica y expansion
    kernel=cv.getStructuringElement(cv.MORPH_ELLIPSE,(25,25))

    closed=cv.morphologyEx(thresh,cv.MORPH_CLOSE,kernel)
    closed=cv.erode(closed,None,iterations=4)
    closed=cv.dilate(closed,None,iterations=4)
```

```
    return closed
#en este apartado, encontramos el contorno del area
def findcnts_and_box_point(closed):
    (cnts,_)=cv.findContours(closed.copy(), cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
    c=sorted(cnts,key=cv.contourArea, reverse=True)[0]
    #computariza la rotacion del la caja a lo largo del contorno
    rect=cv.minAreaRect(c)
    box=np.int0(cv.boxPoints(rect))

    return box
def drawcnts_and_cut(original_img,box):
    #dibuja una caja alrededor de la imagen cuando detecte los bordes
    draw_img=cv.drawContours(original_img.copy(),[box],-1,(0,0,255),3)

    Xs=[i[0] for i in box]
    Ys=[i[1] for i in box]
    x1=min(Xs)
    x2=max(Xs)
    y1=min(Ys)
    y2=max(Ys)
    hight=y2-y1
    width=x2-x1
    crop_img=original_img[y1:y1+hight, x1:x1+width]

    return draw_img,crop_img

def walk():
    #agragamos la direccion donde se encuentra la imagen a analizar
    #y la direccion donde guardaremos la nueva imagen
    img_path=r'C:/Users/andre/Pictures/Inteligencia Artificial/imagen43.jpg' #43.jpg
    save_path=r'C:/Users/andre/Pictures/Inteligencia Artificial/Resultados/imagen9_save.jpeg'
    #asignamos valores para cada una de las funciones, en este caso
    #la imagen original y en escala de grises
    original_img=get_image(img_path)
    grayimage=rgb2gray(original_img)
    #asignamos la imagen en escala de gris para su proceso en el
    #filtro gaussiano
    gaussimage=gauss(grayimage)
    #asignamos variable para su deteccion de bordes
    mag=sobel(gaussimage,70) #valor de threshold (umbral) va de 0-255
    mag_v=sobel_v(gaussimage,70)
    mag_h=sobel_h(gaussimage,70)
    #binarizacion de la imagen
    #thresh=Thresh_and_blur(mag)
    binario=Thresh_and_blur(mag)#gaussimage,
    #iteracionesd de los bordes para mejora del contorno y forma
    closed=image_morphology(binario)
```

```
    #marca de contorno y corte de imagen
    box=findcnts_and_box_point(closed)
    draw_img,crop_img=drawcnts_and_cut(original_img,box)

    #visualizacion de los resultados
    cv.imshow("Imagen Original",original_img)
    cv.imshow("Imagen en Escala de Grises",grayimage)
    cv.imshow("Filtro Gaussiano de la Image Gray",gaussimage)
    cv.imshow("Bordes Sobel Vertical",mag_v)
    cv.imshow("Bordes Sobel Horizontal",mag_h)
    cv.imshow("Bordes de Sobel",mag)
    cv.imshow('Imagen Binarizada',binario)
    cv.imshow('Imagen con Iteraciones',closed)
    cv.imshow('Imagen con Contornos',draw_img)
    cv.imshow('Imagen Crop',crop_img)
    #finalizacion del proceso
    cv.waitKey(0)
    cv.destroyAllWindows()
    cv.imwrite(save_path,crop_img)
walk()
```

Filtro Anisotrópico

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 13 00:31:29 2021

@author: andre
"""

import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import skimage.io as io
import skimage.filters as flt
%matplotlib inline
# since we can't use imports
import numpy as np
import scipy.ndimage.filters as flt
import warnings
import cv2 as cv

def anisodiff(img,niter=1,kappa=50,gamma=0.1,step=(1.,1.),sigma=0, option=1,ploton=False):
        """
        Anisotropic diffusion.

        Usage:
        imgout = anisodiff(im, niter, kappa, gamma, option)

        Arguments:
                img    - input image
                niter  - number of iterations
                kappa  - conduction coefficient 20-100 ?
                gamma  - max value of .25 for stability
                step   - tuple, the distance between adjacent pixels in (y,x)
                option - 1 Perona Malik diffusion equation No 1
                         2 Perona Malik diffusion equation No 2
                ploton - if True, the image will be plotted on every iteration

        Returns:
                imgout   - diffused image.

        kappa controls conduction as a function of gradient.  If kappa is low
        small intensity gradients are able to block conduction and hence diffusion
        across step edges.  A large value reduces the influence of intensity
        gradients on conduction.

        gamma controls speed of diffusion (you usually want it at a maximum of
        0.25)
```

```
step is used to scale the gradients in case the spacing between adjacent
pixels differs in the x and y axes

Diffusion equation 1 favours high contrast edges over low contrast ones.
Diffusion equation 2 favours wide regions over smaller ones.

Reference:
P. Perona and J. Malik.
Scale-space and edge detection using ansotropic diffusion.
IEEE Transactions on Pattern Analysis and Machine Intelligence,
12(7):629-639, July 1990.

Original MATLAB code by Peter Kovesi
School of Computer Science & Software Engineering
The University of Western Australia
pk @ csse uwa edu au
<http://www.csse.uwa.edu.au>

Translated to Python and optimised by Alistair Muldal
Department of Pharmacology
University of Oxford
<alistair.muldal@pharm.ox.ac.uk>

June 2000  original version.
March 2002 corrected diffusion eqn No 2.
July 2012 translated to Python
"""

# ...you could always diffuse each color channel independently if you
# really want
if img.ndim == 3:
        warnings.warn("Only grayscale images allowed, converting to 2D matrix")
        img = img.mean(2)

# initialize output array
img = img.astype('float32')
imgout = img.copy()

# initialize some internal variables
deltaS = np.zeros_like(imgout)
deltaE = deltaS.copy()
NS = deltaS.copy()
EW = deltaS.copy()
gS = np.ones_like(imgout)
gE = gS.copy()

# create the plot figure, if requested
if ploton:
```

```
        import pylab as pl
        from time import sleep

        fig = pl.figure(figsize=(20,5.5),num="Anisotropic diffusion")
        ax1,ax2 = fig.add_subplot(1,2,1),fig.add_subplot(1,2,2)

        ax1.imshow(img,interpolation='nearest')
        ih = ax2.imshow(imgout,interpolation='nearest',animated=True)
        ax1.set_title("Original image")
        ax2.set_title("Iteration 0")

        fig.canvas.draw()

for ii in np.arange(1,niter):

        # calculate the diffs
        deltaS[:-1,: ] = np.diff(imgout,axis=0)
        deltaE[: ,:-1] = np.diff(imgout,axis=1)

        if 0<sigma:
                deltaSf=flt.gaussian_filter(deltaS,sigma);
                deltaEf=flt.gaussian_filter(deltaE,sigma);
        else:
                deltaSf=deltaS;
                deltaEf=deltaE;

        # conduction gradients (only need to compute one per dim!)
        if option == 1:
                gS = np.exp(-(deltaSf/kappa)**2.)/step[0]
                gE = np.exp(-(deltaEf/kappa)**2.)/step[1]
        elif option == 2:
                gS = 1./(1.+(deltaSf/kappa)**2.)/step[0]
                gE = 1./(1.+(deltaEf/kappa)**2.)/step[1]

        # update matrices
        E = gE*deltaE
        S = gS*deltaS

        # subtract a copy that has been shifted 'North/West' by one
        # pixel. don't as questions. just do it. trust me.
        NS[:] = S
        EW[:] = E
        NS[1:,:] -= S[:-1,:]
        EW[:,1:] -= E[:,:-1]

        # update the image
        imgout += gamma*(NS+EW)

        if ploton:
```

```
                    iterstring = "Iteration %i" %(ii+1)
                    ih.set_data(imgout)
                    ax2.set_title(iterstring)
                    fig.canvas.draw()
                    # sleep(0.01)

        return imgout

def anisodiff3(stack,niter=1,kappa=50,gamma=0.1,step=(1.,1.,1.),option=1,ploton=False):
        """
        3D Anisotropic diffusion.

        Usage:
        stackout = anisodiff(stack, niter, kappa, gamma, option)

        Arguments:
              stack  - input stack
              niter  - number of iterations
              kappa  - conduction coefficient 20-100 ?
              gamma  - max value of .25 for stability
              step   - tuple, the distance between adjacent pixels in (z,y,x)
              option - 1 Perona Malik diffusion equation No 1
                       2 Perona Malik diffusion equation No 2
              ploton - if True, the middle z-plane will be plotted on every
                   iteration

        Returns:
              stackout   - diffused stack.

        kappa controls conduction as a function of gradient.  If kappa is low
        small intensity gradients are able to block conduction and hence diffusion
        across step edges.  A large value reduces the influence of intensity
        gradients on conduction.

        gamma controls speed of diffusion (you usually want it at a maximum of
        0.25)

        step is used to scale the gradients in case the spacing between adjacent
        pixels differs in the x,y and/or z axes

        Diffusion equation 1 favours high contrast edges over low contrast ones.
        Diffusion equation 2 favours wide regions over smaller ones.

        Reference:
        P. Perona and J. Malik.
        Scale-space and edge detection using ansotropic diffusion.
        IEEE Transactions on Pattern Analysis and Machine Intelligence,
        12(7):629-639, July 1990.
```

Original MATLAB code by Peter Kovesi
School of Computer Science & Software Engineering
The University of Western Australia
pk @ csse uwa edu au
<http://www.csse.uwa.edu.au>

Translated to Python and optimised by Alistair Muldal
Department of Pharmacology
University of Oxford
<alistair.muldal@pharm.ox.ac.uk>

June 2000 original version.
March 2002 corrected diffusion eqn No 2.
July 2012 translated to Python
"""

```
# ...you could always diffuse each color channel independently if you
# really want
if stack.ndim == 4:
        warnings.warn("Only grayscale stacks allowed, converting to 3D matrix")
        stack = stack.mean(3)

# initialize output array
stack = stack.astype('float32')
stackout = stack.copy()

# initialize some internal variables
deltaS = np.zeros_like(stackout)
deltaE = deltaS.copy()
deltaD = deltaS.copy()
NS = deltaS.copy()
EW = deltaS.copy()
UD = deltaS.copy()
gS = np.ones_like(stackout)
gE = gS.copy()
gD = gS.copy()

# create the plot figure, if requested
if ploton:
        import pylab as pl
        from time import sleep

        showplane = stack.shape[0]//2

        fig = pl.figure(figsize=(20,5.5),num="Anisotropic diffusion")
        ax1,ax2 = fig.add_subplot(1,2,1),fig.add_subplot(1,2,2)

        ax1.imshow(stack[showplane,...].squeeze(),interpolation='nearest')
```

```
            ih =
ax2.imshow(stackout[showplane,...].squeeze(),interpolation='nearest',animated=True)
            ax1.set_title("Original stack (Z = %i)" %showplane)
            ax2.set_title("Iteration 0")

            fig.canvas.draw()

        for ii in np.arange(1,niter):

            # calculate the diffs
            deltaD[:-1,: ,:  ] = np.diff(stackout,axis=0)
            deltaS[:  ,:-1,: ] = np.diff(stackout,axis=1)
            deltaE[:  ,: ,:-1] = np.diff(stackout,axis=2)

            # conduction gradients (only need to compute one per dim!)
            if option == 1:
                gD = np.exp(-(deltaD/kappa)**2.)/step[0]
                gS = np.exp(-(deltaS/kappa)**2.)/step[1]
                gE = np.exp(-(deltaE/kappa)**2.)/step[2]
            elif option == 2:
                gD = 1./(1.+(deltaD/kappa)**2.)/step[0]
                gS = 1./(1.+(deltaS/kappa)**2.)/step[1]
                gE = 1./(1.+(deltaE/kappa)**2.)/step[2]

            # update matrices
            D = gD*deltaD
            E = gE*deltaE
            S = gS*deltaS

            # subtract a copy that has been shifted 'Up/North/West' by one
            # pixel. don't as questions. just do it. trust me.
            UD[:] = D
            NS[:] = S
            EW[:] = E
            UD[1:,: ,: ] -= D[:-1,:  ,:  ]
            NS[: ,1:,: ] -= S[:  ,:-1,:  ]
            EW[: ,: ,1:] -= E[:  ,:  ,:-1]

            # update the image
            stackout += gamma*(UD+NS+EW)

            if ploton:
                iterstring = "Iteration %i" %(ii+1)
                ih.set_data(stackout[showplane,...].squeeze())
                ax2.set_title(iterstring)
                fig.canvas.draw()
                # sleep(0.01)

        return stackout
```

```
img=io.imread('C:/Users/andre/Pictures/Inteligencia Artificial/imagen50.jpg')
img=img.astype(float)
#img=img[300:600,300:600]
m=np.mean(img)
s=np.std(img)
nimg=(img-m)/s
plt.imshow(nimg)
#plt.colorbar()
plt.figure(figsize=(16,9))
fimg=anisodiff(nimg,100,80,0.075,(1,1),2.5,1)
plt.subplot(2,3,1)
plt.imshow(nimg)
plt.title('Original')
plt.subplot(2,3,2)
plt.imshow(fimg,vmin=-1,vmax=1)
#plt.imshow(fimg)
plt.title('Filtered')
plt.subplot(2,3,3)
plt.imshow(fimg-nimg)
plt.title('Difference')
plt.subplot(2,3,4)
h=np.histogram(nimg,100)
plt.plot(h[0])

plt.subplot(2,3,5)
h,ax=np.histogram(fimg,100)

plt.plot(ax[0:(np.size(h))],h)
```

Red Neuronal:

Clase: Entrenamiento

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 26 20:34:06 2021

@author: andre
"""

#importamos librerias necesarias
import sys
from PIL import Image
sys.modules['Image']=Image
import os
from tensorflow.keras import optimizers #libreria optimizadores para entreanimiento de modelo
from tensorflow.keras.preprocessing.image import ImageDataGenerator #Libreria preprocesa
imagenes
from tensorflow.keras.layers import Dropout, Flatten, Dense,Activation #li
from tensorflow.keras.layers import Convolution2D, MaxPooling2D #Capas donde hacemos
convoluciones
from tensorflow.keras.models import Sequential #liberria para redes neuronales secuenciales
from tensorflow.keras import backend as k
from tensorflow.keras import applications
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential

k.clear_session() #limpiamos por si hubiera algun otro proceso corriendo
datosEntrenamiento= './data/entrenamiento'
datosValidacion= './data/validacion'

longitud, altura= 150,150 #long y altura de las imagenes
#reescalamos las imagenes para que los valores en vez de ir de 1 a 255 vayan de 0 a 1
entDatagen=ImageDataGenerator(rescale=1./255)

#en el grupo de validacion simplemente reescalamos
validacionDatagen= ImageDataGenerator(rescale=1./255)

Entrenamiento= entDatagen.flow_from_directory(datosEntrenamiento,
                                target_size=(altura, longitud),
batch_size=32,class_mode='categorical')
Validacion= validacionDatagen.flow_from_directory(datosValidacion,
                                    target_size=(altura,longitud),
                                    batch_size=32, class_mode='categorical')

#creamos red convolucional (modelo preentrenado de google)
vgg=applications.vgg16.VGG16()
```

```
#vgg=applications.vgg16.VGG16()# Ejemplo del modelo
model=Sequential()
#cargamos las capas del modelo preentrenado del modelo secuencial  y se quita la ultima capa
for capa in vgg.layers:
   model.add(capa)
model.layers.pop()
#solo se quiere que entrene la capa que se esta añadiendo
for layer in model.layers:
   layer.trainable=False

#añadimos nuestra propia capa
model.add(Dense(256,activation='softmax'))

#Compilamos el modelo
model.compile(loss='categorical_crossentropy',
        optimizer=optimizers.Adam(lr=0.0004),
        metrics=['accuracy'])


#utilizamos la funcion v para entrenar el algoritmo
model.fit(Entrenamiento,
                steps_per_epoch=100,
                epochs=4,
                validation_data=Validacion,
                validation_steps=300)


#Finalizado el entrenamiento se guarda el modelo
model.save('./modelo/modelo.h5')
model.save_weights('./modelo/pesos.h5')
```

Clase: Entrenar

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 26 20:34:22 2021

@author: andre
"""

import sys
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Flatten, Dense, Activation
from tensorflow.keras.layers import Convolution2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import backend as K
from tensorflow.keras.models import Sequential

K.clear_session()

data_entrenamiento='./data/entrenamiento'
data_validacion='./data/validacion'

#Parametros
epocs=3
longitud,altura=100,100
batch_size=3
pasos=100 #numero de veces que procesa en cada epoca
pasos_validacion=300
filtrosConv1=32
filtrosConv2=64
tamano_filtro1=(3,3)
tamano_filtro2=(2,2)
tamano_pool=(2,2)
clases=3#carpetas a analizar
lr=0.0004 # ajustes para acercarse a la solucion optima

#pre procesamiento de imagenes
entrenamiento_datagen= ImageDataGenerator(
    rescale=1./255, #cada pixel tiene rango de 0-255
    shear_range=0.2, #genera imagen inclinadas
    zoom_range=0.2, #Zoom
    horizontal_flip=True #inversion de imagen
    )
validacion_datagen=ImageDataGenerator(
    rescale=1./255
```

```
    )
imagen_entrenamiento= entrenamiento_datagen.flow_from_directory(
    data_entrenamiento,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical',
    )
imagen_validacion=validacion_datagen.flow_from_directory(
    data_validacion,
    target_size=(altura,longitud),
    batch_size=batch_size,
    class_mode='categorical'
    )

#crear la red CNN
model=Sequential()

model.add(Convolution2D(filtrosConv1, tamano_filtro1, padding='same',
input_shape=(altura,longitud,3), activation='relu'))

model.add(MaxPooling2D(pool_size=tamano_pool))

model.add(Convolution2D(filtrosConv2, tamano_filtro2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=tamano_pool))

model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(clases, activation='softmax'))

model.compile(loss='categorical_crossentropy',
          optimizer=optimizers.Adam(lr=lr),
          metrics=['accuracy'])

model.fit(imagen_entrenamiento,
        steps_per_epoch=pasos,
        epochs=epocs,
        validation_data=imagen_validacion,
        validation_steps=pasos_validacion)


dir='./modelo/'

if not os.path.exists(dir):
    os.mkdir(dir)
model.save('./modelo/modelo.h5')
model.save_weights('./modelo/pesos.h5')
```

Clase: Adivinar

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 26 20:33:30 2021

@author: andre
"""

import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.python.keras.models import Sequential #libreria para redes neuronales
secuenciales
from tensorflow.keras.models import load_model
from tensorflow.python.keras import applications
from tensorflow.python.keras.layers import Dense

longitud, altura=224, 224
weights_model='./modelo/pesos.h5'
vgg=applications.vgg16.VGG16()
cnn=Sequential()
for capa in vgg.layers:
    cnn.add(capa)
cnn.layers.pop()
for layer in cnn.layers:
    layer.trainable=False
cnn.add(Dense(2,activation='softmax'))

cnn.load_weights(weights_model)

def predict(file):
    x=load_img(file,target_size=(longitud,altura))
    x=img_to_array(x)
    x=np.expand_dims(x,axis=0)
    array=cnn.predict(x)
    result=array[0]
    answer=np.argmax(result)

    if answer==0:
        print("Mancha Foliar")
    elif answer==1:
        print("Gusano Trozador")


    return answer
```

Clase: Predecir

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 26 20:34:35 2021

@author: andre
"""

import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.models import load_model

longitud,altura=100,100
modelo='./modelo/modelo.h5'
pesos='./modelo/pesos.h5'
cnn=load_model(modelo)
cnn.load_weights(pesos)

def predict(file):
    x=load_img(file,target_size=(longitud,altura))
    x=img_to_array(x)
    x=np.expand_dims(x, axis=0)
    arreglo=cnn.predict(x) ##[[1,0,0]]
    resultado=arreglo[0] ##[1,0,0]
    respuesta=np.argmax(resultado) #2

    if respuesta==0:
        print("Planta de Maiz")
    elif respuesta==1:
        print("Mancha Foliar")
    elif respuesta==2:
        print("Gusano Trozador")


predict('prueba3.jpg')#Ruta y nombre del archivo a predecir
```