

TECNOLÓGICO NACIONAL DE MEXICO

Instituto Tecnológico de Culiacán



TITULACIÓN INTEGRAL

OPCION:

TESIS PROFESIONAL

NOMBRE DEL PROYECTO:

**Conteo de hojas para la optimización de la producción de un
huerto de jitomate**

ALUMNO(S)

Lizbeth Margarita Armenta Bojórquez

CARRERA:

Ingeniería en sistemas computacionales

Num(s). De Control.

15171071

Culiacán, Sinaloa 2024

DIVISION DE ESTUDIOS PROFESIONALES
ASUNTO: Autorización de Impresión

Culiacán, Sinaloa, 17/10/2024 9:56

C. LIZBETH MARGARITA ARMENTA BOJORQUEZ
PASANTE(S) DE LA CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES
P R E S E N T E.

De acuerdo con fallo emitido por la Comisión Revisora Integrada por:

Cédula Profesional

Presidente:	Maestría en Ciencias en Ciencias de la Computación Gloria Ekaterine Peralta Peñuñuri	6038275
Secretario:	Doctorado en Ciencias en Ingeniería Eléctrica Opción en Sistemas Computacionales Héctor Rodríguez Rangel	9972835
Vocal:	Maestría en Ciencias en Ciencias Computacionales Arcelia Judith Bustillos Martínez	13118410

Y considerando que reúne los requisitos establecidos en la Opción: TITULACION INTEGRAL: TESIS PROFESIONAL "CONTEO DE HOJAS PARA LA OPTIMIZACIÓN DE LA PRODUCCIÓN DE UN HUERTO DE JITOMATE", damos a usted nuestra autorización para que proceda a imprimir su trabajo profesional.

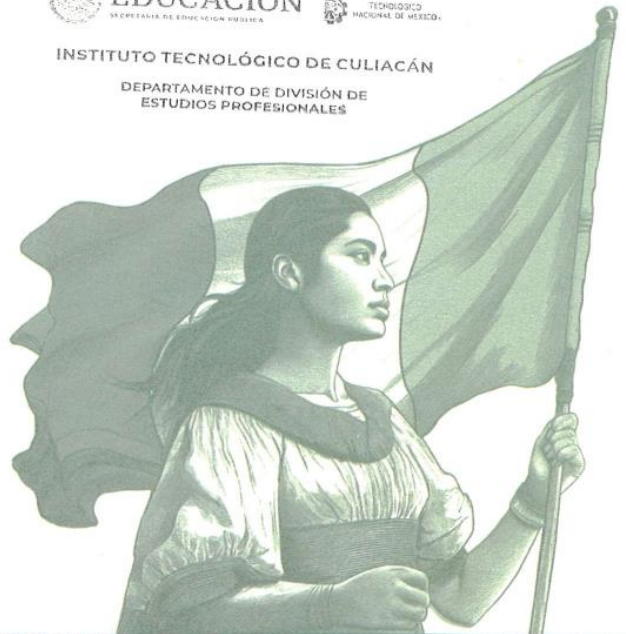
Sin otro particular, quedo de usted.

ATENTAMENTE

Excelencia en Educación Tecnológica®


LIC. CONCEPCIÓN MENDOZA ROSALES
JEFA DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

**EDUCACIÓN**
SECRETARÍA DE EDUCACIÓN PÚBLICA
**TECNOLÓGICO NACIONAL DE MÉXICO**
INSTITUTO TECNOLÓGICO DE CULIACÁN
DEPARTAMENTO DE DIVISIÓN DE ESTUDIOS PROFESIONALES



Agradecimiento

Quiero expresar mi más sincero agradecimiento a mis padres y a mi familia por su apoyo incondicional a lo largo de toda mi vida. Su amor y aliento han sido fundamentales en cada paso de mi camino. También me gustaría agradecer a mis compañeros por los momentos valiosos que hemos compartido. Cada uno de nosotros ha contribuido al aprendizaje y crecimiento mutuo, y estoy profundamente agradecido por ello. En especial quiero agradecer a mis amigos Sofia, Karina, Miguel, Sanjuan y Juan Luis de los que he aprendido tantas cosas, por su apoyo incondicional y sus consejos. Y a mis asesores Gloria Ekaterine Peralta y Héctor Rodríguez Rangel por su constante apoyo y retroalimentación. Su orientación ha sido crucial para mi desarrollo y mejora continua.

Conteo de hojas para la optimización de la producción de un huerto de jitomate

Resumen

El presente documento detalla las actividades realizadas para el desarrollo de un sistema que permita realizar el conteo de hojas de plantas de jitomate el cual servirá para realizar una poda adecuada del cultivo de jitomate para la optimización de su crecimiento evitando así la pérdida de nutrientes. En el desarrollo del proyecto se utilizó un Dataset con 1000 imágenes de hojas de jitomate, las cuales fueron etiquetadas utilizando LabelImg en el formato de Yolo. El desarrollo del código se realizó en Python en la plataforma de Google Colab.

Contenido

1	Introducción	1
1.1	Antecedentes.....	1
1.2	Descripción del problema	1
1.3	Hipótesis o preguntas de investigación	2
1.4	Objetivo general.....	2
1.5	Objetivos específicos.....	2
1.6	Justificación	3
1.7	Alcance	3
2	Fundamento teórico.....	4
2.1	Red neuronal artificial	4
2.2	Red neuronal convolucional cnn.....	5
2.3	Capa convolución	6
2.4	Capa de reducción.....	7
2.5	Clasificador	8
2.6	Algoritmo you only look once (YOLO).....	8
2.7	Arquitectura de YOLO	9
2.8	Darknet	10
2.9	Detección de objetos	11
2.10	Tensorflow	11
2.11	TensorFlow lite	11
2.12	Google Colab.....	12
2.13	Python	13
2.14	Labellmg.....	13

2.15	Cultivo de jitomate	14
2.16	Jitomate	14
2.17	Área foliar (IAF)	14
2.18	¿Por qué se realiza la poda en tomate?	15
3	Actividades realizadas	16
3.1	Capacitación	16
3.2	Descripción de la empresa	17
3.3	Obtención de requisitos	18
3.4	Análisis	18
3.5	Desarrollo	19
3.5.1	Creación del dataset.....	19
3.5.2	Entrenamiento de la red neuronal	21
3.5.3	Convertir el modelo darknet a tensorflow lite.....	24
3.5.4	Imprimir el número de hojas detectadas en la imagen	24
3.6	Pruebas	25
4	Resultados.....	26
5	Conclusiones	35
6	Competencias desarrolladas	36
7	Fuentes de información	37

Índice de figuras y tablas

Figura 1: Estructura de las capas de una red neuronal [2]	5
Figura 2: Capas de una red neuronal convolucional [3].	6
Figura 3: Capa convolucional [3]	7
Figura 4: Capa de reducción [3]	8
Figura 5: Funcionamiento de yolo [5]	9
Figura 6: Arquitectura de YOLO [6]	10
Figura 7: Clasificador de imágenes	16
Figura 8: Detector de pájaros	17
Figura 9: Métricas de rendimiento [19]	19
Figura 10: Etiquetación de imágenes de hojas de jitomate numero 1	20
Figura 11: Etiquetación de imágenes de hojas de jitomate numero 2	20
Figura 12: Comando para exportar dataset.....	21
Figura 13: Comando para clonar repositorio de darknet	21
Figura 14: Comando de entrenamiento	21
Figura 15: Entrenamiento de la red	22
Figura 16: Resultado del entrenamiento numero 1.....	23
Figura 17: Resultado del entrenamiento numero 2.....	23
Figura 18: Comando para convertir un archivo.weights a TensorFlow.....	24
Figura 19: Método cv2.putText()	25
Figura 20: Conteo de hojas número 1	27
Figura 21: Conteo de hojas número 2	28
Figura 22: Conteo de hojas número 3	29
Figura 23: Conteo de hojas número 4	30
Figura 24: Conteo de hojas número 5	31
Figura 25: Conteo de hojas número 6	32
Figura 26: Conteo de hojas número 7	33
Figura 27: Conteo de hojas número 8	34

Tabla 1: Requerimientos funcionales	18
Tabla 2: Conteos de hojas.....	34

1 Introducción

Las plantas de jitomate con mayor área foliar y ambiente favorable realizan una fotosíntesis más eficiente, factor que aumenta el rendimiento. Para conocer el área foliar de una planta de jitomate es necesario saber el número de hojas, su velocidad de crecimiento y su tamaño final.

Realizar el conteo de hojas manualmente puede ser muy tardado es por lo que se necesita un contador automático de hojas para hacer más rápido el conteo de las hojas de jitomate.

Para la detección de las hojas se utilizó YOLO ya que es una de las técnicas más rápidas y se puede utilizar en tiempo real en video. YOLO utiliza una red neuronal convolucional llamada Darknet para clasificar imágenes y agrega una parte de detección, un cuadrado con posiciones x e y, altura y ancho del objeto encontrado.

1.1 Antecedentes

El proceso de la poda de las hojas de las plantas de jitomate es muy importante que se realice periódicamente ya que esto garantiza que los jitomates sean de mayor tamaño y calidad. Ocasionando como inconveniente un aumento de mano de obra para realizar este proceso de poda que llega a ser tardado por el conteo de las hojas de forma manual a la planta del jitomate.

Con el desarrollo de este proyecto se busca disminuir la mano de obra que se necesita para realizar el conteo de las hojas y mejorar el conteo de las hojas al ser más preciso el conteo, así como también reducir los tiempos para realizar el conteo.

1.2 Descripción del problema

El tiempo en que se realiza el conteo de hojas de plantas de jitomate es lento, debido a que las personas realizan el conteo de hojas manualmente y deben que realizar el conteo más de una vez para comprobar que es correcto el número de hojas.

Conteo incorrecto de las hojas de plantas de jitomate, debido a que se realiza manualmente lo que ocasiona que se confundan las hojas contadas con las no contadas debido a que no se van marcando al momento en el que se realiza el conteo de las hojas de plantas de jitomate.

1.3 Hipótesis o preguntas de investigación

El diseño de un sistema para el conteo de hojas en plantas de jitomate permitirá realizar un conteo más preciso, lo cual dará como resultado la obtención de una mejor gestión del cultivo de jitomate y una mayor productividad de las plantas al tener datos más precisos.

1.4 Objetivo general

Diseñar un sistema que realice el conteo de hojas de plantas de jitomate para la optimización de su crecimiento. El conteo de las hojas de plantas de jitomate se realizará mediante un cuaderno de Google Colab entrenando la red neuronal con una GPU Tesla T4 utilizando un Dataset con más de 1000 fotos de hojas de Jitomate y el detector diminuto YOLOv4. Al finalizar el entrenamiento por medio de una fotografía de la planta se realizará el conteo de las hojas de la planta.

1.5 Objetivos específicos

- Hacer más rápido el proceso de conteo de hojas de plantas de jitomate.
- Obtener un conteo de hojas más preciso.
- Generar un Dataset para el entrenamiento de la red neuronal.
- Realizar el entrenamiento de la red neuronal utilizando el Dataset de hojas de jitomate.

1.6 Justificación

Este proyecto es originado para la optimización de la producción de un huerto de jitomate, en base al número de hojas por planta. Las plantas con mayor área foliar y ambiente favorable son capaces de realizar una fotosíntesis más eficiente, factor que aumenta el rendimiento de los cultivos.

Para conocer el área foliar de una planta es necesario saber el número de hojas, su velocidad de crecimiento y su tamaño final. Actualmente el conteo de hojas se realiza manualmente lo cual es muy tardado, debido a que periódicamente se tiene que estar realizando el conteo con el objetivo de llevar un control del comportamiento del número de hojas a través del tiempo, porque a medida que se desarrolla la planta de jitomate, las hojas se vuelven más complejas y funcionales.

Con este proyecto se busca facilitar el conteo de las hojas mediante la utilización de inteligencia artificial, creando una aplicación que realice el conteo de las hojas por medio de imágenes de las plantas, para saber cuál de las densidades de población de plantas alcanza el mayor número de hojas y rendimiento en el cultivo de jitomate.

1.7 Alcance

Desarrollar un contador de hojas de jitomate a partir de imágenes de plantas de jitomate utilizando Google Colab. El contador de hojas está diseñado exclusivamente para trabajar con fotografías, no siendo compatible con videos. Para una buena precisión del contador es necesario contar con fotografías de buena calidad ya que en fotografías de baja calidad no es posible realizar una buena detección de las hojas de las plantas.

2 Fundamento teórico

En esta parte del documento se muestran los temas y conceptos relevantes utilizados para poder comprender y llevar a cabo el desarrollo del sistema, así como las herramientas utilizadas.

2.1 Red neuronal artificial

Las redes neuronales artificiales (también conocidas como sistemas conexionistas) son un modelo computacional vagamente inspirado en el comportamiento observado en su homólogo biológico. Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida [1].

El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque las redes neuronales son más abstractas. Las redes neuronales actuales suelen contener desde unos miles a unos pocos millones de unidades neuronales [2].

El funcionamiento de las redes se asemeja al del cerebro humano. Las redes reciben una serie de valores de entrada y cada una de estas entradas llega a un nodo llamado neurona. Las neuronas de la red están a su vez agrupadas en capas que forman la red neuronal. Cada una de las neuronas de la red posee a su vez un peso, un valor numérico, con el que modifica la entrada recibida. Los nuevos valores obtenidos salen de las neuronas y continúan su camino por la red. Este funcionamiento puede observarse de forma esquemática en la figura 1 [2].

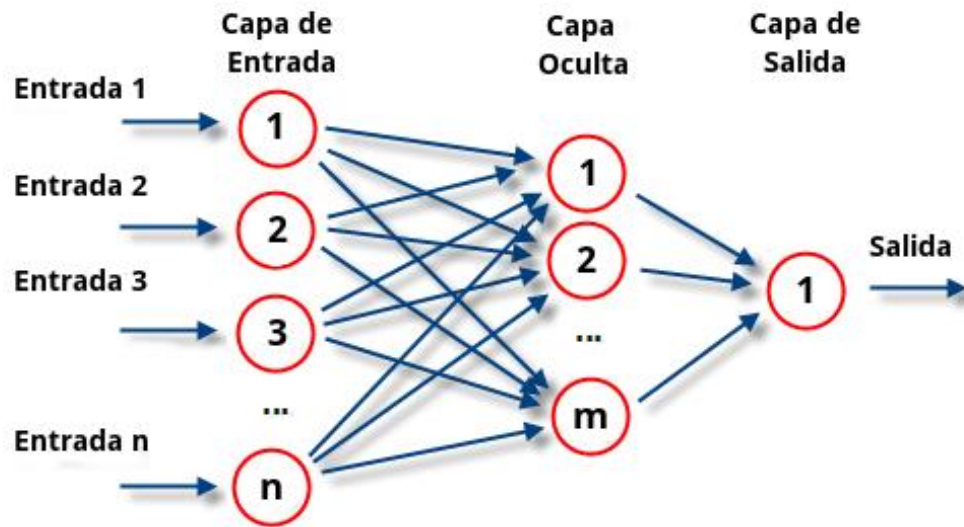


Figura 1: Estructura de las capas de una red neuronal [2]

2.2 Red neuronal convolucional cnn

Las redes neuronales convolucionales son similares a las redes neuronales multicapa, su principal ventaja es que cada parte de la red se le entrena para realizar una tarea, esto reduce significativamente el número de capas ocultas, por lo que el entrenamiento es más rápido. Además, presenta invarianza a la traslación de los patrones a identificar [3].

Las redes neuronales convolucionales son muy potentes para todo lo que tiene que ver con el análisis de imágenes, debido a que son capaces de detectar características simples como por ejemplo detección de bordes, líneas, etc. y componer en características más complejas hasta detectar lo que se busca, en la figura 2 se observan las capas de la red neuronal convolucional [3].

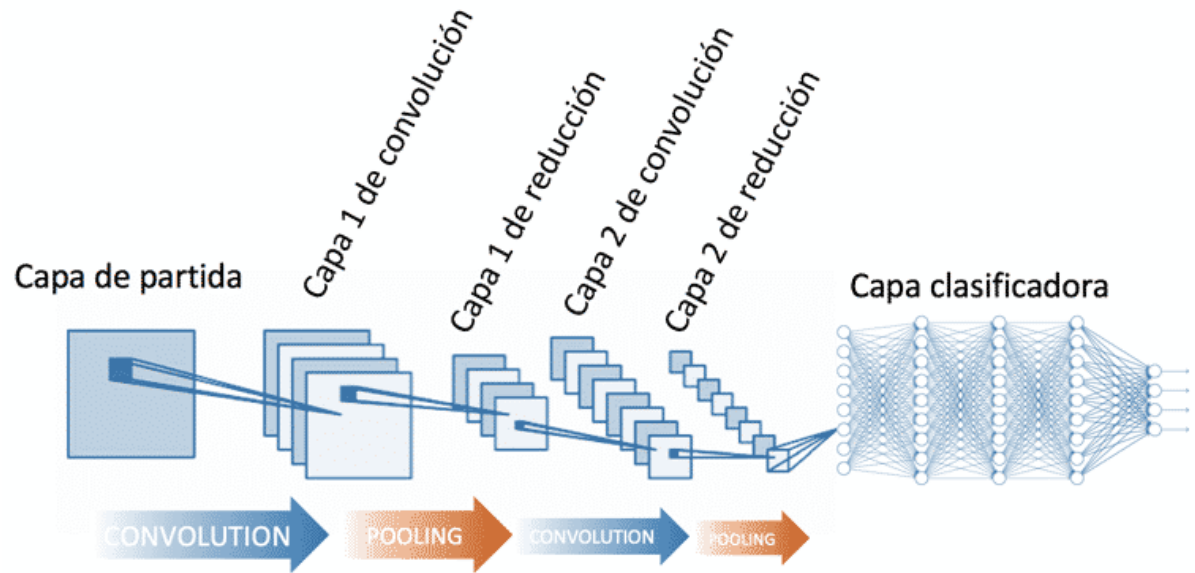


Figura 2: Capas de una red neuronal convolucional [3].

2.3 Capa convolución

En la convolución se realizan operaciones de productos y sumas entre la capa de partida y los n filtros (o kernel) que genera un mapa de características. Las características extraídas corresponden a cada posible ubicación del filtro en la imagen original [3].

La ventaja es que el mismo filtro (= neurona) sirve para extraer la misma característica en cualquier parte de la entrada, con esto que consigue reducir el número de conexiones y el número de parámetros a entrenar en comparación con una red multicapa de conexión total, en la figura 3 se observa la capa convolucional [3].

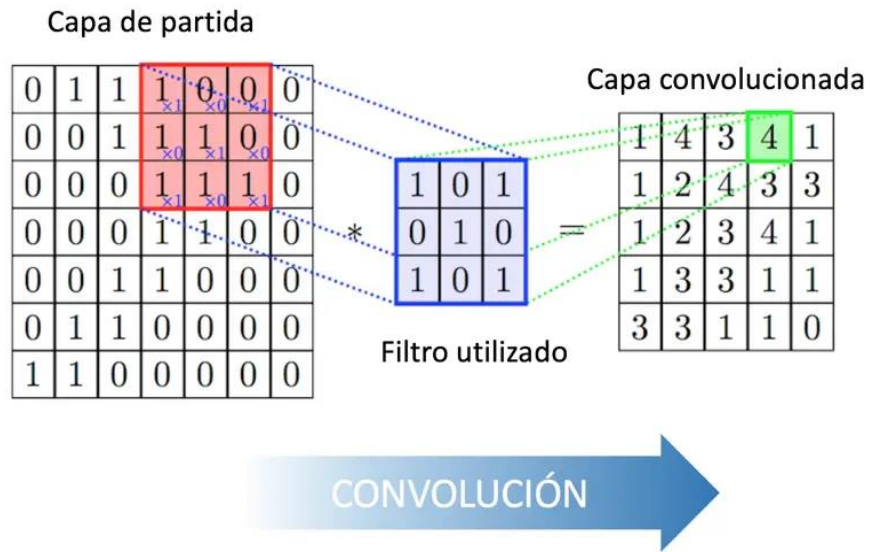


Figura 3: Capa convolucional [3]

2.4 Capa de reducción

En la reducción se disminuye la cantidad de parámetros al quedarse con las características más comunes [3].

La forma de reducir parámetros se realiza mediante la extracción de estadísticas como el promedio o el máximo de una región fija del mapa de características, al reducir características el método pierde precisión, aunque mejora su compatibilidad, en la figura 4 se observa la capa de reducción [3].

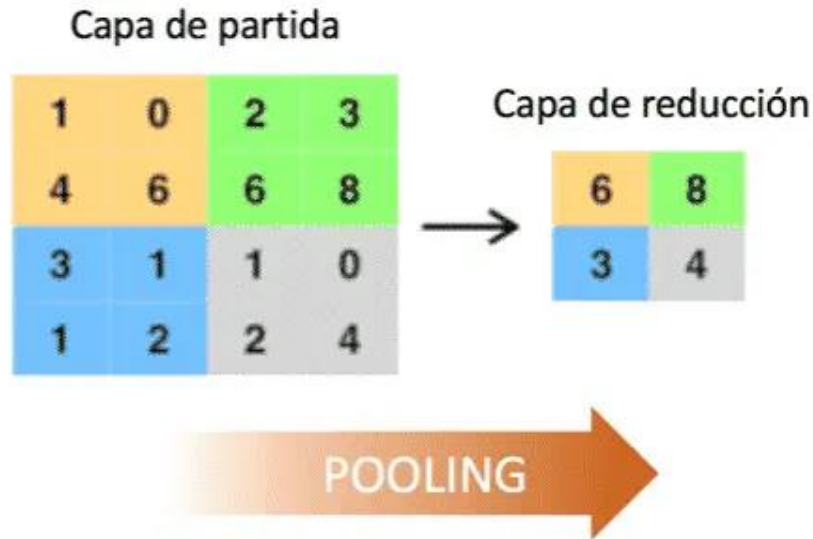


Figura 4: Capa de reducción [3]

2.5 Clasificador

El final de las capas convolucional y de reducción, se suele utilizar capas completamente conectadas en la que cada píxel se considera como una neurona separada al igual que en un perceptrón multicapa [3].

La última capa de esta red es una capa clasificadora que tendrá tantas neuronas como el número de clases a predecir [3].

2.6 Algoritmo you only look once (YOLO)

El algoritmo You Only Look Once (YOLO), es un sistema de código abierto del estado del arte para detección de objetos en tiempo real, el cual hace uso de una única red neuronal convolucional para detectar objetos en imágenes. Para su funcionamiento, la red neuronal divide la imagen en regiones, prediciendo cuadros de identificación y probabilidades por cada región; las cajas son ponderadas a partir de las probabilidades predichas. El algoritmo aprende representaciones generalizables de los

objetos, permitiendo un bajo error de detección para entradas nuevas, diferentes al conjunto de datos de entrenamiento [4].

Para llevar a cabo la detección primero divide la imagen en una cuadrícula de $S \times S$ (figura 5 imagen de la izquierda). En cada una de las celdas predice N posibles “bounding boxes” y calcula el nivel de certidumbre (o probabilidad) de cada una de ellas (figura 5 imagen del centro), es decir, se calculan $S \times S \times N$ diferentes cajas, la gran mayoría de ellas con un nivel de certidumbre muy bajo. Después de obtener estas predicciones se procede a eliminar las cajas que estén por debajo de un límite. A las cajas restantes se les aplica un paso de “non-max suppression” que sirve para eliminar posibles objetos que fueron detectados por duplicado y así dejar únicamente el más exacto de ellos (figura 5 imagen de la derecha) [5].

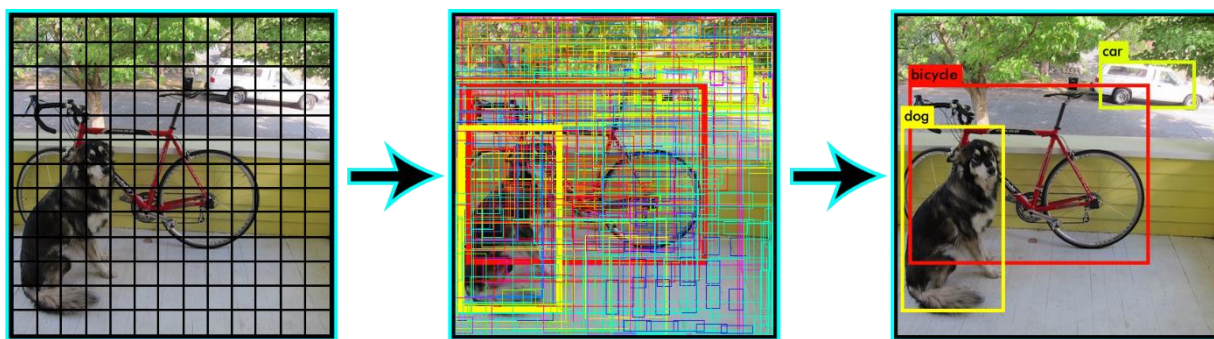


Figura 5: Funcionamiento de yolo [5]

2.7 Arquitectura de YOLO

El modelo se implementó como una red neuronal convolucional y fue evaluado en el set de datos para detección de PASCAL VOC. Las capas convolucionales iniciales de la red se encargan de la extracción de características de la imagen, mientras que las capas de conexión completa predicen la probabilidad de salida y las coordenadas del objeto [4].

La red tiene 24 capas convolucionales seguidas por 2 capas de conexión completa; esta hace uso de capas de reducción de 1×1 seguidas capas convolucionales de 3×3 . El modelo Fast YOLO hace uso de

una red neuronal de 9 capas. La salida final del modelo tensor de predicción de 7x7x30, la figura 6 muestra la arquitectura de YOLO [4].

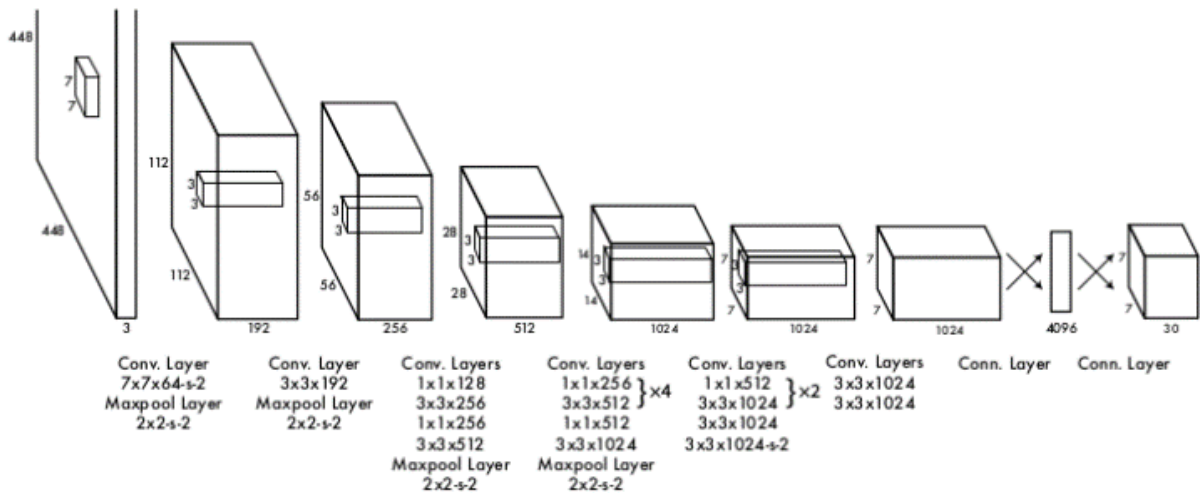


Figura 6: Arquitectura de YOLO [6]

2.8 Darknet

Darknet es un framework o entorno para redes neuronales de código abierto escrito en C y CUDA. Es rápido, fácil de instalar y soporta computación de CPU y GPU [7].

Entre las funcionalidades que nos ofrece este entorno, destacan las redes neuronales YOLO (You Only Look Once, sólo miras una vez) que se caracterizan por aplicar una sola red neuronal a toda la imagen, la cual divide la imagen en regiones más pequeñas, luego predice las cajas delimitadoras y asigna probabilidades para cada una de estas regiones. Éste hecho hace que la inferencia en este tipo de estructuras sea más rápida que en otro tipo de estructuras de redes convencionales [7].

Proporciona distintas funcionalidades, entre las que encontramos:

- darknet.exe detector: Es el script principal e implementa las siguientes funcionalidades en función del segundo parámetro que introduzcamos:
 - train: Entrenamiento

- valid: Validación
- demo: Nos permite ejecutar la red sobre un archivo de video o en tiempo real utilizando una cámara.
- cfg: Carpeta con plantillas de configuraciones sobre distintas redes.
- scripts: Carpeta con diferentes scripts que nos serán de utilidad, como por ejemplo gen_anchors.py, el cual utiliza K-Means para calcular de forma óptima las dimensiones de los anchors que utilizarán las distintas capas de la red [7].

2.9 Detección de objetos

La detección de objetos es una tecnología de ordenador relacionada con la visión artificial y el procesamiento de imagen que trata de detectar casos de objetos semánticos de una cierta clase (como humanos, edificios, o coches) en vídeos e imágenes digitales. Los ámbitos mejor desarrollados de detección de objetos incluyen detección de caras y detección de personas. La detección de objetos tiene aplicaciones en muchas áreas de visión artificial, incluyendo recuperación de imágenes y video vigilancia [8].

2.10 Tensorflow

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que les permite a los investigadores impulsar un aprendizaje automático innovador y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de AA [9].

2.11 TensorFlow lite

TensorFlow Lite es un conjunto de herramientas para ayudar a los desarrolladores a ejecutar modelos de TensorFlow en dispositivos incorporados, móviles o de IoT. Permite la inferencia de

aprendizaje automático en dispositivos con una latencia baja y un tamaño de objeto binario pequeño [10].

2.12 Google Colab

Google Colaboratory o Google Colab es probablemente el servicio en la nube más popular para el aprendizaje automático. Como parte del área de Google Cloud, ofrece acceso gratuito a la informática de GPU y TPU y permite un excelente ecosistema para trabajar y desarrollar algoritmos. Una buena alternativa al trabajo a nivel local [11].

Google Colab es también uno de los servicios en la nube más famosos para científicos de datos experimentados, investigadores e ingenieros de software [11].

Si bien Google Colaboratory parece fácil de comenzar, tiene algunos aspectos de más complejidad. Dentro de todo, hay varias ventajas de usar Google Colaboratory, por sobre el trabajo a nivel local con el uso de las propias máquinas locales [11].

Algunos de los beneficios concretos de Google Colab son:

- No necesita hacer una configuración de entorno. Viene con paquetes importantes preinstalados y listos para usar.
- Proporciona acceso directo en el navegador a Jupyter Notebook
- GPU gratis
- Permite almacenar cuadernos en Google Drive
- Permite importar cuadernos desde Github
- Entrega un código de documento con Markdown
- Da la opción de cargar datos desde la unidad

2.13 Python

Es un lenguaje de programación interpretado, multiparadigma y multiplataforma usado, principalmente, en Big Data, AI (Inteligencia Artificial), Data Science, frameworks de pruebas y desarrollo web. Esto lo convierte en un lenguaje de propósito general de gran nivel debido a su extensa biblioteca, cuya colección ofrece una amplia gama de instalaciones [12].

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones [13].

Python usa tipado dinámico y conteo de referencias para la administración de memoria [13].

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos) [13].

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable [13].

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme [13].

2.14 Labellmg

Labellmg es una herramienta de anotación de imágenes gráficas. Está escrito en Python y usa Qt para su interfaz gráfica. Las anotaciones se guardan como archivos XML en formato PASCAL VOC, el formato utilizado por ImageNet. Además, también es compatible con el formato YOLO [14].

2.15 Cultivo de jitomate

El Jitomate constituye en nuestro país una de las hortalizas más importantes debido a la cantidad de empleos directos e indirectos que genera su cultivo y al número de divisas que ingresan al país por concepto de su comercialización [15].

Debido a la superficie destinada a su cultivo y al valor de la producción, el jitomate (*Solanum lycopersicon* L.) es uno de los cultivos más importantes de México pues ocupa el décimo lugar a nivel mundial en superficie sembrada, esta especie hortícola es la más cultivada tanto a cielo abierto como en Invernadero [15].

2.16 Jitomate

El jitomate es uno de los cultivos más importantes de México y del mundo, tanto por su importancia económica como por ser fuente de vitaminas, minerales y antioxidantes, Los minerales que contiene son calcio, fósforo, potasio y sodio y las vitaminas que contiene son A, B1, B2, y C [16].

Además, tiene propiedades medicinales entre las que destacan las siguientes: antiséptico, alcalinizante, depurativo, diurético, digestivo, laxante, desinflamatorio y remineralizante [16].

2.17 Área foliar (IAF)

El índice de área foliar (IAF) es la expresión numérica adimensional resultado de la división aritmética del área de las hojas de un cultivo expresado en m^2 y el área de suelo sobre el cual se encuentra establecido, también expresado en m^2 . El IAF permite estimar la capacidad fotosintética de las plantas y ayuda a entender la relación entre acumulación de biomasa y rendimiento bajo condiciones ambientales imperantes en una región determinada [17].

El IAF puede estimarse por diferentes métodos, algunos de estos se describen brevemente a continuación: Métodos destructivos: consiste en la toma de muestras representativas del cultivo y se contabiliza el área foliar por medio de ceptómetros (integrador de área electrónico). Cálculo empírico

del IAF: Tomar dos plantas por unidad de estudio, medir y multiplicar largo por ancho de cada hoja, el resultado multiplicarlo por el factor 0.75, sumar los valores obtenidos de cada hoja por planta, promediar resultados de cada planta. Posteriormente determinar el área de suelo ocupada por planta. Finalmente el IAF será determinado dividiendo el área foliar de la planta entre el área de suelo ocupada por esta [17].

2.18 ¿Por qué se realiza la poda en tomate?

La poda de hojas o deshojado, es una práctica que consiste en remover las hojas senescentes inferiores (hojas viejas o dañadas) por debajo del último racimo que va madurando, dejando un racimo adicional descubierto. Con el deshojado se consigue una mejor ventilación, uniformidad en la coloración de los frutos y mayor eficiencia en la aplicación de agroquímicos, por lo tanto, el deshoje se debe realizar de manera periódica, procurando no quitar más de tres hojas a la vez para evitar un desbalance energético e hídrico que agote a la planta y repercuta en el rendimiento del mismo [18].

La poda de brotes y hojas debe ser limpia a ras de tallo para evitar el ataque de enfermedades como Botrytis. Existen dos métodos para llevar a cabo estas prácticas: el mecánico, que consiste en emplear tijeras desinfectadas (con yodo, permanganato de potasio, hipoclorito de sodio o sales cuaternarias de amonio) para evitar el ataque de patógenos; y el método manual, donde se da un tirón hacia arriba aprovechando el punto de unión entre tallo y hoja [18].

3 Actividades realizadas

A continuación, se enlistan las actividades realizadas para desarrollar el sistema de conteo de hojas de plantas de jitomate

3.1 Capacitación

Para el desarrollo del sistema fue necesario realizar un curso con el Dr. Rodríguez Rangel Héctor donde se abordaron los temas de redes neuronales (RNA), perceptrón, perceptrón multicapa, redes recurrentes (RNN) y Long Short Term Memory (LSTM) y se realizó una red neuronal utilizando Tensorflow 2.0.

Además, se abordó el tema de redes convolucionales y se realizó un ejemplo de clasificación de imágenes utilizando las redes convolucionales como se muestra en la figura 7. También se realizó un detector de pájaros utilizando Yolo como se muestra en la figura 8.

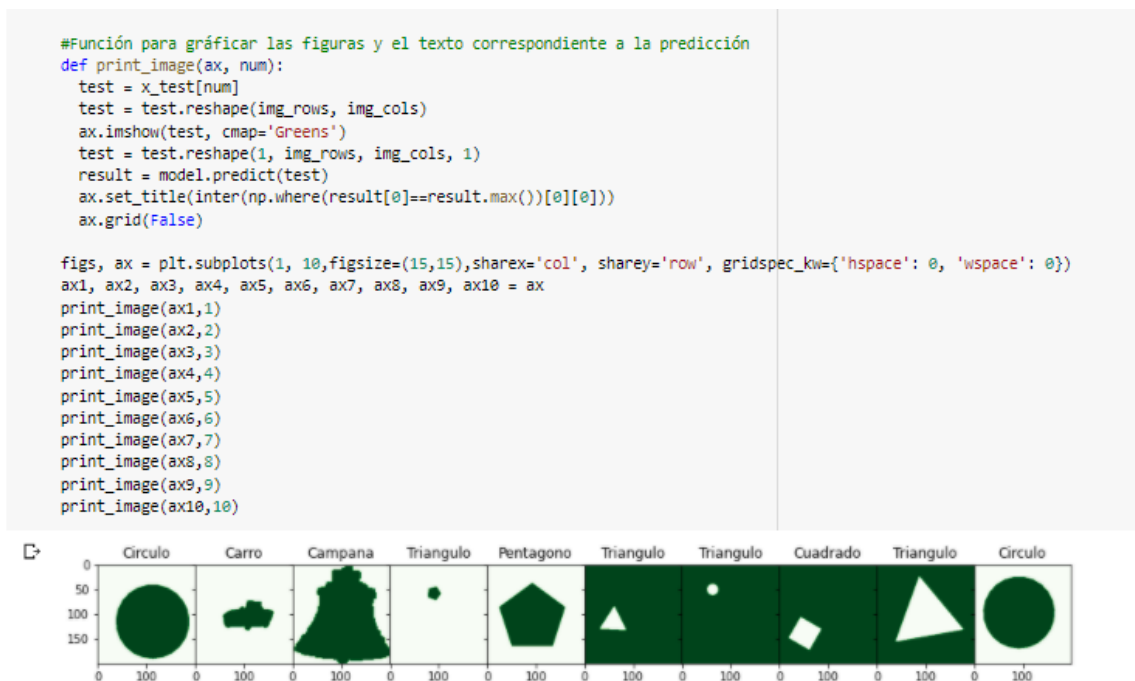


Figura 7: Clasificador de imágenes

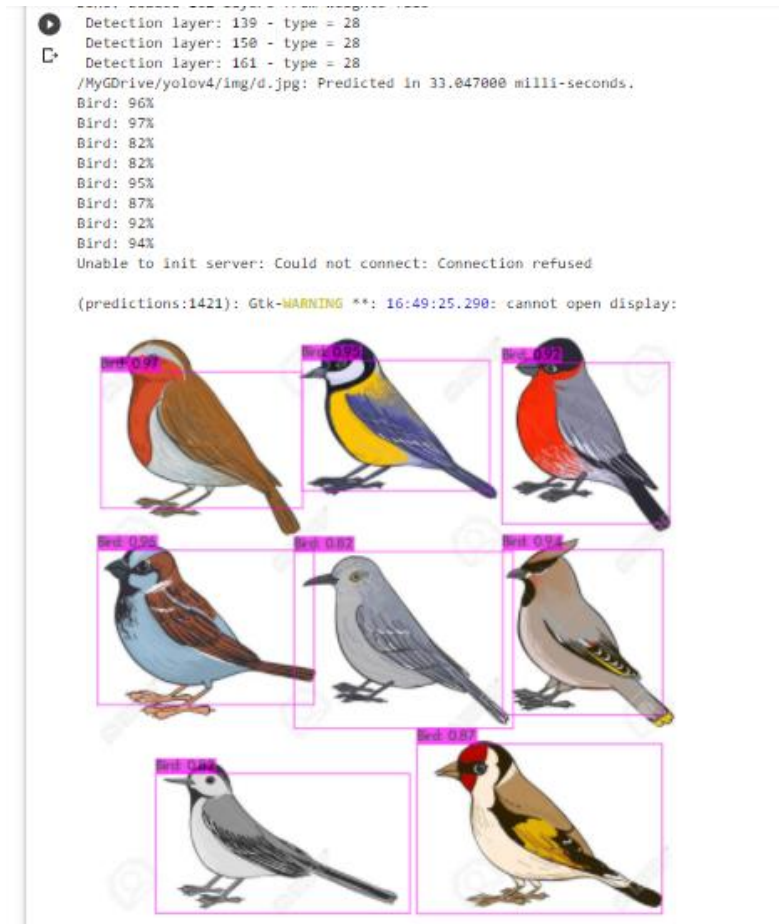


Figura 8: Detector de pájaros

3.2 Descripción de la empresa

El Tecnológico Nacional de México, Campus Culiacán (o Instituto Tecnológico de Culiacán) inició labores en septiembre de 1968. El ITC oferta opciones educativas en las áreas de ingeniería y tecnología. El compromiso, la capacidad de análisis con sentido crítico, la responsabilidad y el proponer soluciones adecuadas a las necesidades regionales, fueron y siguen siendo hoy en día, algunos de los principios básicos de la formación profesional que se ofrece.

La zona de influencia del Instituto Tecnológico Regional de Culiacán abarcaba los municipios de Angostura, Badiraguato, Concordia, Cosalá, Elota, Escuinapa, Mazatlán, Mocorito, El Rosario, Salvador Alvarado y San Ignacio.

Las funciones sustantivas del Instituto Tecnológico de Culiacán son la docencia, la investigación y la extensión de la cultura.

3.3 Obtención de requisitos

En esta actividad se realizó una recolección de información de lo que la empresa necesitaba, llevando a cabo reuniones para así obtener las especificaciones requeridas para el proyecto. La tabla 1 muestra los requerimientos funcionales solicitados por la empresa.

Tabla 1: Requerimientos funcionales

Requerimientos funcionales	
Clave	Descripción
RF01	El sistema debe ser capaz de detectar las hojas de jitomate en imágenes.
RF02	El sistema debe de contar el número de hojas detectadas.

3.4 Análisis

En esta etapa, se examinó la mejor forma posible de cumplir con lo solicitado y así cumplir con los objetivos. Analizando las métricas de rendimiento se observó que YOLOv4 tiny es aproximadamente 8 veces más rápido en el momento de la inferencia que YOLOv4 y aproximadamente 2/3 del rendimiento en MS COCO como se muestra en la figura 9. Es por lo que se decidió realizar el entrenamiento de la red utilizando YOLOv4 tiny.

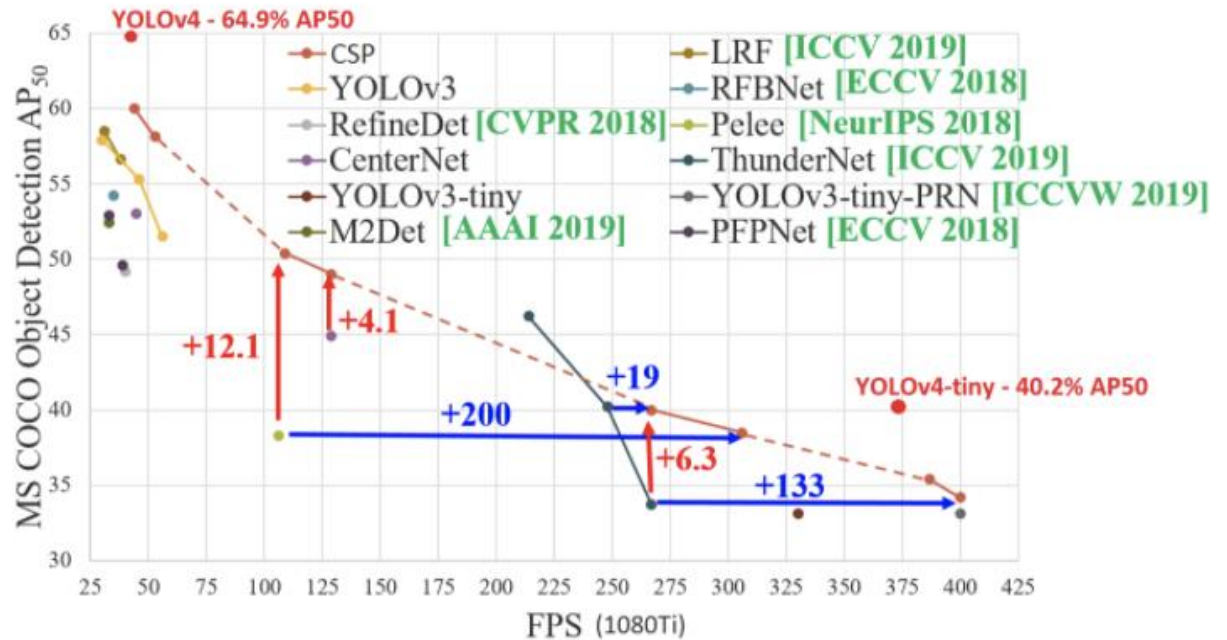


Figura 9: Métricas de rendimiento [19]

3.5 Desarrollo

A continuación, se muestran las actividades realizadas durante el desarrollo del sistema de conteo de hojas de jitomate.

3.5.1 Creación del dataset

La creación del dataset consta de la recolección de fotos de hojas de jitomate tomadas en diferentes posiciones y agrupaciones, se recolectaron más de 1000 fotos de hojas de jitomate.

Para el etiquetado del dataset se utilizó la herramienta LabelImg ya que es una de las herramientas de anotación de imágenes más fácil de utilizar y es compatible con el formato YOLO.

En las figuras 10 y 11 se muestra la etiquetación de un conjunto de hojas de jitomate utilizando la herramienta LabelImg, donde las hojas se encierran en un rectángulo para identificar su posición. En la

figura 10 se muestra un conjunto de ramificaciones de hojas de jitomate que se encuentran en diferentes posiciones y tamaños etiquetadas individualmente y en la figura 11 se muestra una hoja de jitomate.



Figura 10: Etiquetación de imágenes de hojas de jitomate numero 1

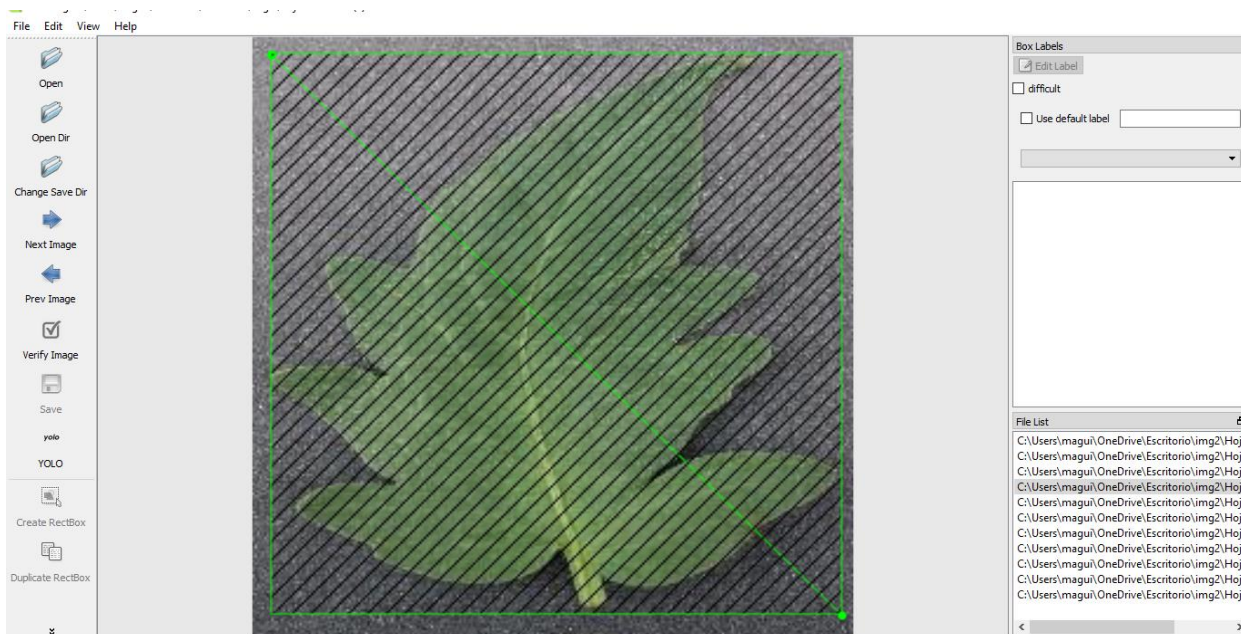


Figura 11: Etiquetación de imágenes de hojas de jitomate numero 2

3.5.2 Entrenamiento de la red neuronal

Para realizar el entrenamiento de la red neuronal se dividió el dataset 70% para entrenamiento, 20% para validación y 10% para pruebas. Por cuestiones de seguridad se almaceno el dataset en un repositorio de GitHub.

En un cuaderno de Google Colab entrenando con una GPU Tesla T4, se exportó el dataset utilizando el comando que se observa en la figura 12.

```
[1] #clonar el dataset
!git clone https://github.com/Magui017/dataset

Cloning into 'dataset'...
remote: Enumerating objects: 2017, done.
remote: Counting objects: 100% (2017/2017), done.
remote: Compressing objects: 100% (1988/1988), done.
remote: Total 2017 (delta 0), reused 2017 (delta 0), pack-reused 0
Receiving objects: 100% (2017/2017), 41.08 MiB | 44.89 MiB/s, done.
```

Figura 12: Comando para exportar dataset

Después se clonó el repositorio de darknet mantenido por Roboflow [20] con el comando que se observa en la figura 13.

```
[ ] #clonar el darknet
!git clone https://github.com/roboflow-ai/darknet.git
```

Figura 13: Comando para clonar repositorio de darknet

Se realizó la configuración del modelo porque los modelos YOLO ajustan ligeramente la arquitectura de la red en función del número de clases en su conjunto de datos personalizado. Y la duración de la formación también debe ajustarse en función del número de clases.

Después de tener el entorno, el dataset y la configuración de entrenamiento, se pasó al entrenamiento del detector diminuto YOLOv4 personalizado con el comando que se observa en la figura 14.

```
[ ] !./darknet detector train data/obj.data cfg/custom-yolov4-tiny-detector.cfg yolov4-tiny.conv.29 -dont_show -map
```

Figura 14: Comando de entrenamiento

El entrenamiento de la red se realizó aproximadamente en 40 minutos, terminó en la iteración 2000 con un accuracy del 91% y se creó un archivo .weights. Como se muestra en la figura 15.

```
Last accuracy mAP@0.5 = 91.33 %, best = 91.33 %
1999: 0.096494, 0.189774 avg loss, 0.000026 rate, 0.710439 seconds, 127936 images, 0.020673 hours left
Loaded: 0.000037 seconds

(next mAP calculation at 2000 iterations)
Last accuracy mAP@0.5 = 91.33 %, best = 91.33 %
2000: 0.194427, 0.190239 avg loss, 0.000026 rate, 0.722427 seconds, 128000 images, 0.020468 hours left

calculation mAP (mean average precision)...
200
detections_count = 1071, unique_truth_count = 365
class_id = 0, name = hoja, ap = 91.26% (TP = 324, FP = 81)

for conf_thresh = 0.25, precision = 0.80, recall = 0.89, F1-score = 0.84
for conf_thresh = 0.25, TP = 324, FP = 81, FN = 41, average IoU = 64.83 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.912588, or 91.26 %
Total Detection Time: 1 Seconds

Set -points flag:
`-points 101` for MS COCO
`-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
`-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset

mean_average_precision (mAP@0.5) = 0.912588
Saving weights to backup//custom-yolov4-tiny-detector_2000.weights
Saving weights to backup//custom-yolov4-tiny-detector_last.weights
Saving weights to backup//custom-yolov4-tiny-detector_final.weights
```

Figura 15: Entrenamiento de la red

Con el archivo .weights se realizaron pruebas para comprobar el buen funcionamiento de la red, utilizando las imágenes de la carpeta de test se realizaron las pruebas aleatoriamente obteniendo buenos resultados como se muestra en las figuras 16 y 17.



Figura 16: Resultado del entrenamiento numero 1

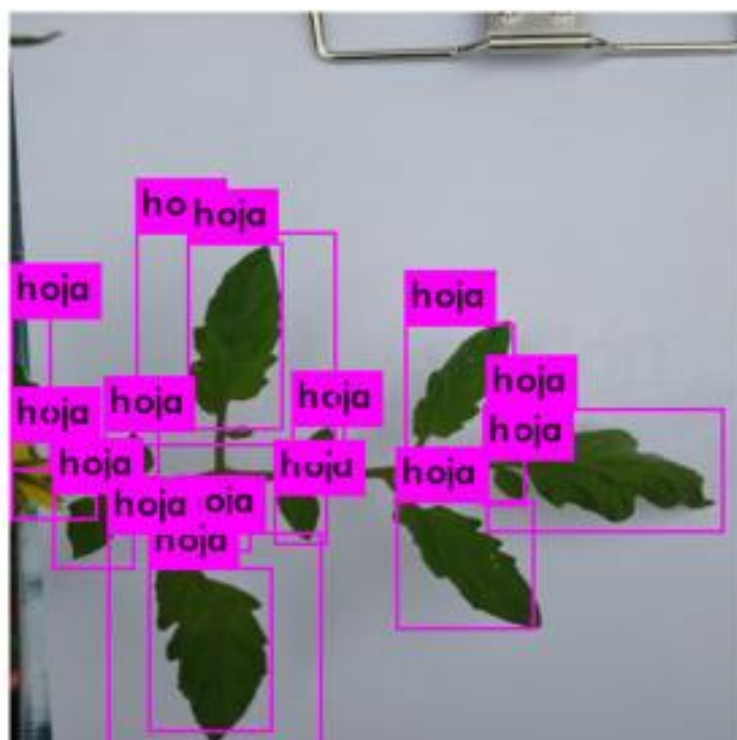


Figura 17: Resultado del entrenamiento numero 2

Se guardó el archivo .weights en el drive para utilizarlo para convertirlo a TensorFlow con el comando que se observa en la figura 18.

```
[ ] #save final weights to google drive
    from google.colab import drive
    drive.mount('/content/drive')

Mounted at /content/drive

[ ] # Darknet Weights
    !cp /content/darknet/backup/custom-yolov4-tiny-detector_final.weights "/content/drive/My Drive"

[ ] #coco.names is hardcoded somewhere in the detector
    %cp data/obj.names data/coco.names
```

Figura 18: Comando para convertir un archivo.weights a TensorFlow

3.5.3 Convertir el modelo darknet a tensorflow lite

Para convertir el modelo darknet (archivo .weights) se utilizó el repositorio tensorflow-yolov4-tflite el cual utiliza el conjunto de datos COCO como base por lo cual se cambiaron las clases a la utilizada en el entrenamiento. Lo que permitió convertir de Darknet a TensorFlow y luego de TensorFlow a TensorFlow Lite.

3.5.4 Imprimir el número de hojas detectadas en la imagen

Para imprimir el número de hojas detectadas en las imágenes se utilizó la biblioteca de enlaces de Python OpenCV, esta biblioteca está diseñada para resolver problemas de visión por computadora. Se utilizó el método cv2.putText() para dibujar la cadena de texto en las imágenes, tal como se muestra en la figura 19.



Figura 19: Método cv2.putText()

3.6 Pruebas

En esta etapa se verificó que el detector de hojas funcionara de la manera correcta, se realizaron distintas pruebas en imágenes de hojas de jitomates donde se comprobó una buena detección de las hojas en las imágenes que estaban bien enfocadas, en las imágenes que estaban borrosas era más difícil la detección de las hojas, cuando en la imagen solo estaba una hoja era 100% su detección cuando se aumentaba la cantidad de hojas y la calidad de la imagen disminuía su porcentaje de detección era bajo.

4 Resultados

Los resultados obtenidos en la detección de hojas de jitomate fueron satisfactorios, con una precisión del 91%. En las distintas pruebas realizadas se observó un buen funcionamiento de la detección de las hojas en imágenes bien tomadas, pero en las imágenes borrosas no se pudo detectar todas las hojas debido a que las hojas no son claramente visibles. A continuación, se muestran los resultados obtenidos, en las figuras 20 y 21 se obtuvo un 100% de efectividad debido a que son imágenes bien enfocadas y claras, en la figura 22 se obtuvo un 64.71% debido a que las hojas se encimaron lo que ocasiona que se no se detectaran correctamente, en las figuras 23, 24, 25, 26 y 27 el porcentaje de efectividad varia del 75% al 88.89% ya que las imágenes no son lo suficientemente claras para un 100% de efectividad. En la tabla 2 se muestran los porcentajes de efectividad del conteo de las hojas.

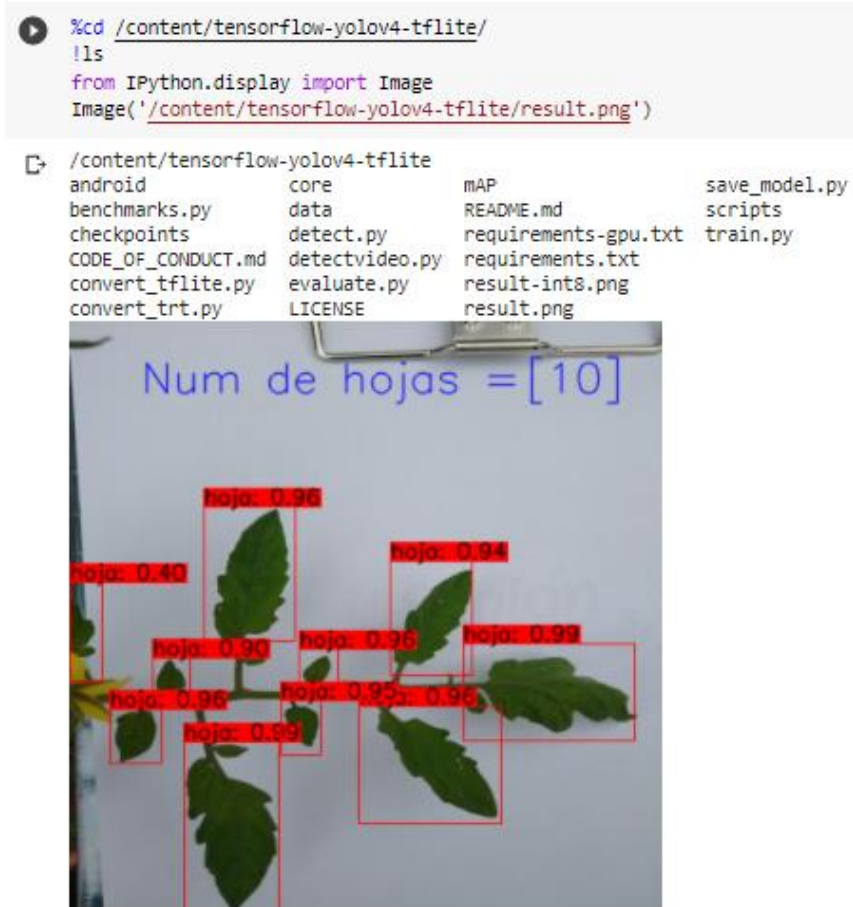


Figura 20: Conteo de hojas número 1



Figura 21: Conteo de hojas número 2



Figura 22: Conteo de hojas número 3



Figura 23: Conteo de hojas número 4



Figura 24: Conteo de hojas número 5

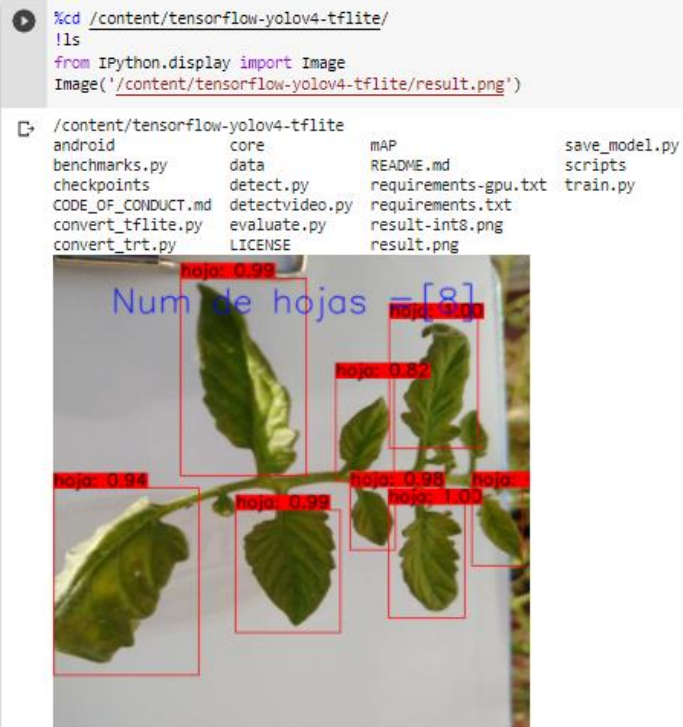


Figura 25: Conteo de hojas número 6



Figura 26: Conteo de hojas número 7



Figura 27: Conteo de hojas número 8

Tabla 2: Conteos de hojas

Num Figura	Total de hojas detectadas	Total de hojas reales	Porcentaje de efectividad
Figura 20	10	10	100
Figura 21	9	9	100
Figura 22	11	17	64.71
Figura 23	22	27	81.48
Figura 24	10	12	83.33
Figura 25	8	9	88.89
Figura 26	8	10	80
Figura 27	6	8	75

5 Conclusiones

En mi experiencia personal, me resultó interesante aprender más sobre el uso de las redes neuronales porque nos permite resolver problemas de la misma manera que el cerebro humano.

Para la realización del proyecto Google Colab fue una herramienta clave para el proyecto gracias a que nos permite el uso de GPU sin necesidad de tener una computadora con GPU, lo que nos permite realizar el entrenamiento en cuestión de minutos.

En el etiquetado de las imágenes de las hojas de jitomate LabelImg facilitó el etiquetado de las imágenes gracias a que es fácil de utilizar.

En el futuro, se puede agregar la detección de plagas, lo que mejorará enormemente el cuidado de las plantas y detectará cualquier plaga a tiempo. Además del hecho de que no sería necesario comprender completamente las diversas plagas que pueden aparecer en las plantas de jitomate para el uso de la aplicación. También recomiendo agregar una función de detección del jitomate para saber la cantidad de jitomates producidos por una planta de jitomate sin necesidad de contarlos manualmente.

6 Competencias desarrolladas

En el transcurso de la realización de este proyecto, fue necesario aplicar las habilidades adquiridas a lo largo de años de estudio, las cuales describiré a continuación:

- Capacidad para utilizar tecnologías de la información y la comunicación.
- Capacidad para buscar, procesar y analizar información de diversas fuentes.
- Capacidad de aplicar los conocimientos en la práctica.
- Capacidad de aprender y actualizar permanentemente.
- Capacidad de investigación.
- Capaz de trabajar de forma autónoma.
- Capacidad para organizar y planificar el tiempo.

Para la realización de la tesis fue fundamental mi capacidad de investigación para encontrar la información necesaria para conocer más sobre la problemática que se presenta cuando los cultivos de jitomate no reciben el cuidado necesario para una buena producción.

Además de mi capacidad para aprender y actualizarme permanentemente lo cual me sirvió para aprender de qué manera funcionan las redes neuronales, la detección de objetos y también me sirvió para aprender a utilizar el LabelImg para la creación del dataset de las fotos de hojas de jitomate.

7 Fuentes de información

1. Colaboradores de Wikipedia. (2020). Red neuronal artificial. Diciembre 20, 2020, de Wikipedia, la enciclopedia libre Sitio web: https://es.wikipedia.org/wiki/Red_neuronal_artificial
2. ATRIA Innovation. (2020). Qué son las redes neuronales y sus funciones. Diciembre 20, 2020, de ATRIA Innovation Sitio web: <https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/>
3. Calvo, D. (2018). Red Neuronal Convolutacional CNN. Diciembre 20, 2020, de diegocalvo.es Sitio web: <https://www.diegocalvo.es/red-neuronal-convolutacional/>
4. Colaboradores de Wikipedia. (2020). Algoritmo You Only Look Once (YOLO). Diciembre 20, 2020, de Wikipedia, la enciclopedia libre Sitio web: [https://es.wikipedia.org/wiki/Algoritmo_You_Only_Look_Once_\(YOLO\)](https://es.wikipedia.org/wiki/Algoritmo_You_Only_Look_Once_(YOLO))
5. A.,E. (2018). Detección de objetos con YOLO: implementaciones y como usarlas. Diciembre 20, 2020, de Medium Sitio web: <https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246>
6. Eesha Ulhaq. (2020). Cómo ven los coches autónomos. diciembre 20, 2020, de ICHI.PRO Sitio web: <https://ichi.pro/es/como-ven-los-autos-sin-conductor-245103176893745>
7. Redmon,J. (2020). Darknet: Open Source Neural Networks in C. Diciembre 20, 2020, de pjreddie Sitio web: <https://pjreddie.com/darknet/>
8. Colaboradores de Wikipedia. (2020). Detección de objetos. Diciembre 20, 2020, de Wikipedia, la enciclopedia libre Sitio web: https://es.wikipedia.org/wiki/Detecci%C3%B3n_de_objetos
9. Tensorflow. (2020). Una plataforma de extremo a extremo de código abierto para el aprendizaje automático. diciembre 20, 2020, de tensorflow Sitio web: <https://www.tensorflow.org/?hl=es-419>
10. TensorFlow. (2020). Guía de TensorFlow Lite. Diciembre 20, 2020, de TensorFlow Sitio web: <https://www.tensorflow.org/lite/guide?hl=es-419>

11. A. (2020). Google Colaboratory Colab - Guía Completa Español. Diciembre 20, 2020, de Marketing Branding Sitio web: <https://www.marketing-branding.com/google-colaboratory-colab-guia-completa-espanol/>
12. Angeles, J. (2020). Python: ¿seguirá siendo el lenguaje de programación más popular en 2021?. Diciembre 20, 2020, de crehana Sitio web: <https://www.crehana.com/mx/blog/web/que-es-python/>
13. Colaboradores de Wikipedia. (2020). Python. Diciembre 20, 2020, de Wikipedia Sitio web: <https://es.wikipedia.org/wiki/Python>
14. tzutalin. (2020). labelImg. Diciembre 20, 2020, de PyPI Sitio web: <https://pypi.org/project/labelImg/>
15. Instituto de Investigación y Capacitación Agropecuaria, Acuícola y Forestal. (2024) Cultivo de Jitomate. <https://icamex.edomex.gob.mx/jitomate>
16. De Agricultura Y Desarrollo Rural, S. (2022). El jitomate, hortaliza mexicana de importancia mundial. gob.mx. <https://www.gob.mx/agricultura/articulos/el-jitomate-hortaliza-mexicana-de-importancia-mundial?idiom=es>
17. Intagri S.C. (2016) El Índice de Área Foliar (IAF) y su Relación con el Rendimiento del Cultivo de Maíz. <https://www.intagri.com/articulos/cereales/el-indice-de-area-foliar-iaf>
18. Intagri S.C. (2016). ¿Por qué se Realiza la poda en Tomate? <https://www.intagri.com/articulos/horticultura-prottegida/por-que-se-realiza-la-poda-en-tomate>
19. Solawetz, J. (2020). Train YOLOv4-tiny on Custom Data - Lightning Fast Object Detection. Diciembre 20, 2020, de Roboflow Blog Sitio web: <https://blog.roboflow.com/train-yolov4-tiny-on-custom-data-lighting-fast-detection>

20. Su, J . (2020). How to Train a Custom Mobile Object Detection Model (with YOLOv4 Tiny and TensorFlow Lite). Diciembre 20, 2020, de Roboflow Sitio web: <https://blog.roboflow.com/how-to-train-a-custom-mobile-object-detection-model/>