

Universidad Nacional de Colombia

Facultad de Ciencias Agrarias

Integración interinstitucional para la búsqueda de sistemas agrícolas sostenibles frente al cambio climático

Procesamiento de imágenes de hojas de aguacate
Reporte Mayo

Quinche, Gabriel
gquinche@unal.edu.co

Serrano, Edison
edserranoc@unal.edu.co

14 de junio de 2022

1. Introducción

El procesamiento de fotografías desde amateur a obtenidas satelitalmente, o con equipos microscópicos, ha visto una explosión en su uso en la última década, diariamente se publican artículos que usan técnicas como las redes neuronales convolucionales, o más generalmente el Deep Learning para descubrir patrones, hacer predicciones e incluso avanzar la teoría de un área; este patrón al parecer solo será replicado en los próximos años con tecnologías prometedoras como los *transformers*. En el siguiente informe presentamos un pequeño marco teórico con énfasis en la intuición de esta área y, a su vez, los avances hacia una técnica de *segmentación* (reconocimiento de objetos) aplicada a hojas de aguacate con el fin de ser clasificadas como enfermas o sanas. El uso de la herramienta por un agrícola podría evitarle la pérdida de un árbol enfermo al realizar alguna aplicación a tiempo, o incluso reconocer qué colores y patrones extraños en una hoja pueden ser producto del ambiente (como heladas, o quemazón) y cuáles corresponden a una enfermedad sin la ayuda de un experto.

2. Objetivo

El objetivo del trabajo es generar un *notebook* (cuaderno con código ejecutable) o librería de Python que implemente el modelo de clasificación de aguacates, acompañado de documentación amplia y atractiva.

2.1. Objetivos específicos

- Hacer un modelo que sirva para la clasificación de hojas marchitas y no marchitas del árbol de aguacate *Hass*
- Aumentar la robustez del modelo al permitir la *segmentación de instancias* reconociendo múltiples hojas por foto y clasificándolas
- Lograr una extrapolación básica entre cantidad de hojas marchitas y necesidad de acción
- Finalmente, documentar el proceso tanto como código como en forma de artículo introduciendo los temas centrales a la clasificación y comparando desde diferentes tipos de segmentación o arquitecturas

3. Avance

En el momento tanto el primer objetivo como el segundo se encuentra avanzados como acá se presenta, y aunque ya existía un modelo precursor para el punto la mayoría del tiempo de trabajo se ha dedicado a la búsqueda de una herramienta más robusta que pudiera afrontar tanto el primero como el segundo, como es Detectron2 y Yolov5.

4. Marco Teórico

La siguiente sección pretende introducir al lector interesado de posiblemente otra área del conocimiento que las ciencias de la computación y la estadística a las técnicas y teoría subyacente de la herramienta a realizar, esto con el fin de que entienda tanto sus capacidades como sus limitaciones y así poder incorporar la herramienta de una forma ideal con sus objetivos, bien sean prácticos o teóricos.

4.1. Red Neuronal

Una red neuronal en su esencia es una red que conecta componentes que se influencian entre sí dependiendo de cierto nivel de activación, el nombre viene de su inspiración biológica, donde por ejemplo

cierta estimulación en nuestro cerebro da por la percepción de peligro desenfoca en cierto reflejo bien sea físico o mental. A pesar de esto una red neuronal profunda en el fondo es la aplicación de una gran cantidad de operaciones matemáticas (especialmente productos de matrices por vectores y filtrados) a cierta *entrada* como una imagen, sonido, medida de temperatura, etc... generando una *salida* que corresponde bien con alguna de las siguientes.

- predicciones (por ejemplo un estado del clima)
- mediciones (nivel de enfermedad dado cierto color de piel)
- recomendaciones (necesidad de aplicar fungicidas)
- incluso replicas (una hoja que podría pasar como otra de la misma especie)

todas estas, sin embargo, en el fondo son para la red tan solo números o una colección de los mismos (vectores), y son análogos a funciones como $f(x^2)$ con una salida para cada entrada, acumulando complejidad y ganándose el término profundas por ser de la forma $f_1(f_2(f_3(\dots + c) + b) + a)$ donde las funciones más dentro del paréntesis se consideran las más profundas y cercanas a la entrada, y las funciones más externas las más cercanas a la salida. A partir de la composición de muchas de estas funciones parametrizadas de una forma adecuada (mediante el entrenamiento) obtenemos para muchas entradas, salidas que consideramos inteligentes y acordes.

4.2. Clasificación

Cuando la posible salida de una red neuronal para un conjunto de datos es tan solo un 1 o un 0, podemos considerar a la red neuronal como una red clasificadora binaria, por ejemplo una R tal que su entrada son imágenes de animales podría llegar a clasificar entre mamíferos y no mamíferos, sin embargo, de manera inicial una red neuronal no tiene conceptos previos del mundo, de hecho es totalmente aleatoria, y el 0 o 1 de la red es puro ruido y no reflejará ningún aspecto intrínseco o útil de la imagen, para que este número sea útil debemos hacer un proceso llamado entrenamiento.

4.3. Entrenamiento

El entrenamiento es un procedimiento realizado sobre redes neuronales o en general modelos donde se ajusta el modelo, o sus parámetros, tal que la salida del mismo siga cierta finalidad, en las redes neuronales el entrenamiento se realiza mediante herramientas de optimización las cuales principalmente buscan minimizar cierto valor. Para pensar en el entrenamiento de una red neuronal se le puede personificar como un robot que quiere llegar a un punto en un mapa, precisamente un punto por ejemplo de máxima altura, este robot sin embargo no puede ver como nosotros sus alrededores, en el espacio matemático estamos rodeados de niebla, y no podemos considerar la "altura" de cierta posición arbitraria al menos de que nos "paremos" exactamente en ese punto, en términos matemáticos al menos de que evaluemos

cierta función. Sin embargo ya que los parámetros de una red neuronal son miles o millones de números, que pueden ir desde muy negativos a muy positivos, desde valores enteros como 1, 10, 1257 a valores decimales como 0.1 0.010 o 0.00001257, las posibilidades de evaluación son en efecto infinitas, por eso mismo en el entrenamiento le damos cierta brújula a nuestro robot, esta le permite al mismo saber una dirección de gran ascenso desde el punto donde esta parado, siguiendo por mucho tiempo a la brújula el robot puede no llegar al punto más alto en el espacio sin embargo si puede llegar a un punto donde en ninguna dirección ascenderá más, esta cima es un *máximo local* y cuando lo maximizado es que tan *exacto* fue el modelo en reconocer cierto objeto, el modelo podrá por fin por su cuenta reconocer al menos objetos en los que ha sido entrenado.

4.4. Búsqueda de máximos o buena exactitud de un modelo

Que tan bueno puede ser cierto resultado dado en el entrenamiento esta limitado por el mecanismo de optimización como por ejemplo el descenso del gradiente, este nos da máximos locales que en general no son malos, en resumen modelos relativamente buenos, sin embargo esta optimización o búsqueda de un punto alto se realiza solo teniendo en cuenta los *datos de entrenamiento* en el fondo la calidad de cierta posición del espacio para el robot podía no solo depender de la altura sino también de la temperatura en cierto lugar, así cuando se pone en testeo al robot, básicamente se le esta haciendo considerar más cualidad del punto en el espacio en el que esta, de nuevo el punto al que llegamos podía ser una buena cima, sin embargo tener un clima muy regular, el *overfitting* puede ser pensado como cuando el robot solo se concentra en la altura, el over esta relacionado en como por ejemplo permitimos que el robot deambule demasiado tiempo, aunque probablemente terminara en una cima alta allí el clima tendera a ser muy malo, el robot se concentró en una tarea a corto plazo *el entrenamiento* y olvido más cualidades que serian importantes en el mundo real *la validación y los conjuntos de datos* De esta manera el procedimiento estándar para una red neuronal es permitirle entrenar cierta cantidad de tiempo no tan prolongado, si bien con la analogía de la búsqueda de una cima que sacrifica otras cualidad, o como la memorización de los objetos, dado que una red neuronal tiene tantos parámetros relacionados a la operación que hace sobre sus imágenes o objeto a analizar, el buscar una cima a toda costa es el fondo buscar parámetros que hacen que seamos exactos lo máximo que podamos con respecto a cierto conjunto de entrenamiento, lo que traduce en categorizar perfectamente sonidos de gatos, o imágenes de aguacates pero para esto lo que podríamos es simplemente guardar espacio en la red neuronal donde copiamos explícitamente las imágenes de dichas cosas a categorizar, y en efecto aunque este proceso no es una copia uno a uno, muchas veces se llegan a modelos que reflejan perfectamente los datos más no el mundo real, por esto se debe evitar el overfitting.

4.5. Leakeage

Una metafora importante para del aprendizaje profundo es el de dar un libro abierto a una maquina y luego dos examenes el libro abierto es el dataset de entrenamiento, de el con suficiente tiempo y neuronas, una red puede aprender o memorizar cualquier información el tiempo de entrenamiento muchas veces es directamente proporcional a la cantidad de paginas del libro que el modelo logra memorizar, el problema

es que como la maquina en si no es inteligente, puede que la primera pagina del libro le enseñe un esquema general para solucionar problemas, y sin embargo la red intentara usar no este esquema general sino simplemente comparar las preguntas que le hacemos con lo que ha memorizado y poner su respuesta, de nuevo es la importancia de evitar permitirle que tenga acceso a tantas paginas la que a veces le permite a la red adquirir información más universal o *generalizable*, bien siendo así si en el libro que damos a la red se encuentra exactamente una pregunta que le haremos en el examen, o es por ejemplo una muy mínima variación hay una probabilidad muy alta de que la red puntué bien al haber memorizado esa pagina, sin embargo, no aprendió de esa pagina no abstrajo su información para usarla en otro problema distinto, sino simplemente encontró algo exactamente igual que ya sabía hacer y lo hizo, una red neuronal graba patrones dentro de sí para responder correctamente, y si es suficientemente profunda como las redes populares de la actualidad tiene suficiente espacio para grabar exactamente la imagen y decir, si veo esta imagen debo hacer *y*, Respecto a este tema en la base de datos del aguacate había dos problemas importantes, uno la contaminación de datos, y dos la repetición de datos. La contaminación se daba cuando una imagen se encontraba en dos dataset distintos, en este caso como estamos clasificando en enfermedad vs no enfermedad tenemos dos datasets, y además solo dos respuestas *y* = 0 o 1. El tener una imagen en dos datasets es una mala práctica porque la red neuronal con gran probabilidad habrá estudiado dicha imagen y decirle que la respuesta es la contraria hace que lo que aprendió de esa imagen no tenga sentido, existen estrategias más robustas relacionadas a la incertidumbre para dicho proceso, sin embargo aunque esto es problemático es peor problema de la repetición de datos. En la repetición de datos el problema es que vamos a adquirir modelos que hacen en cierta medida inútil el proceso de validación y testeo, cualquier red puede lograr aprender un ejemplo concreto y felicitarla por ello es improductivo y nos dará más confianza en un modelo malo, que en campo no va a rendir pues se va a encontrar con imágenes muy distintas que nunca aprendió como abordar. Generar el desarrollo además de otras herramientas o aplicaciones con un modelo incorrecto o a él que aun le faltan datos puede incluso generar más mal que bien, pues una persona al confiar en el modelo puede tomar una decisión equivocada que por su cuenta no hubiera hecho.

4.6. Regularización

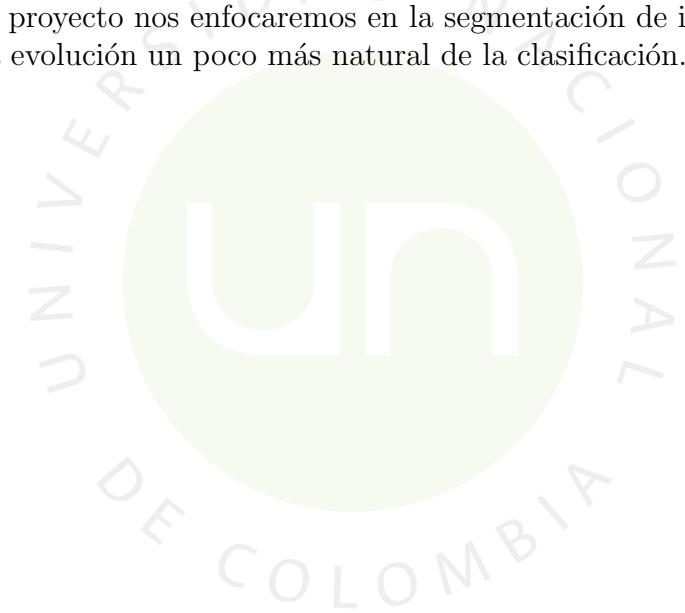
En la estadística tradicional desde hace ya varios años es recomendado el uso de modelos *sparse*, que en general están relacionados a que son simples (tienen muchos ceros respecto a variables no significativas) y aunque generalmente sacrifican algo de precisión en los conjuntos de dato de entrenamiento tienden a tener mejor rendimiento en la validación e incluso en el mundo real, son un ejemplo claro de la idea de la navaja de Ockham donde pensamos que el mejor modelo es uno que es simple, la matemática de como transformar un modelo complejo en un modelo simple pero bueno de hecho no es tan compleja, tiene sus principios computacionales en los años 50 sin embargo no nos dan soluciones analíticas, no existe una fórmula para el modelo sparse de casi ningún objeto, para obtener modelos simples necesitamos usar métodos computacionales conocidos como optimización, los cuales aunque con una teoría matemática profunda han sido transformados al día de hoy en tecnologías,

R, Python, Julia, Matlab

todos cuentas con librerías muy bien documentadas llamadas cvx la mayoría de paquetes de estadística por debajo de cuerdas usan estas técnicas para solucionar problemas conocidos como “textit-LASSO” las redes neuronales no tienen matemática tan sencillas o bien documentadas para

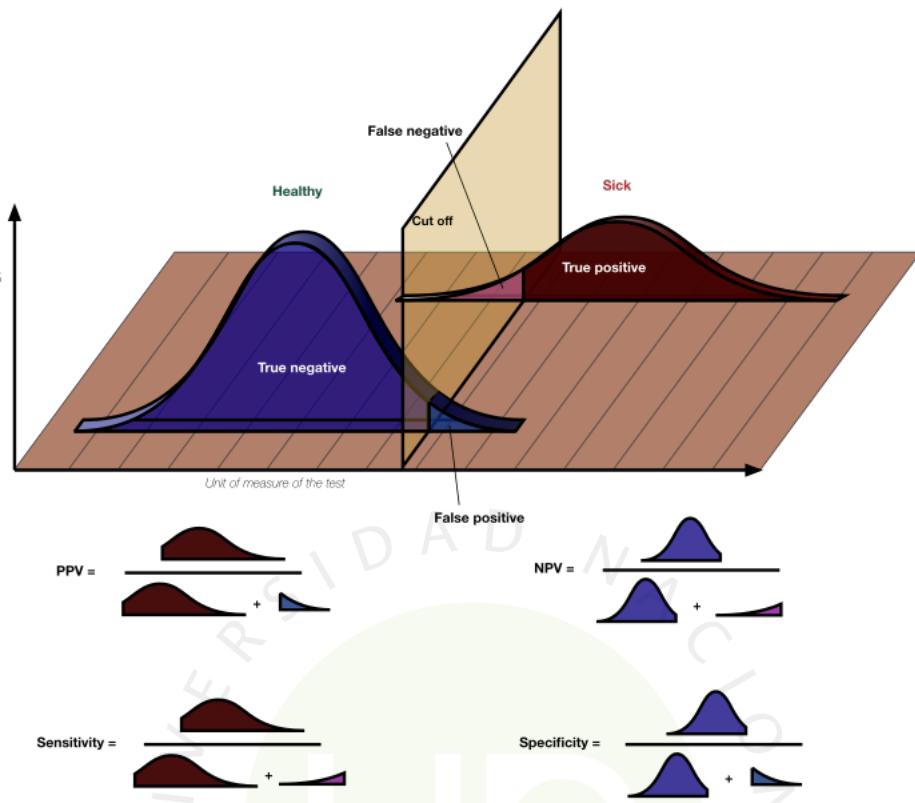
4.7. Tipos de clasificación

La segmentación de un objeto se refiere a la creación de cierta división: partir o designar ciertas regiones que representan una cualidad (por ejemplo el pixel muestra tejido enfermo) la segmentación de instancias ubica en imágenes píxeles relacionados a un objeto en específico, generalmente el objeto a identificar, es decir, identifican objetos individuales como 3 gatos distintos a órganos diferentes como corazón o cada pulmón en una imagen médica, esta identificación se hace en imágenes mediante rectángulos o polígonos que rodean al mismo, cuando usamos rectángulos este problema es denominada *detección de objetos*¹ la segmentación semántica es la agrupación no de objetos (o instancias) individuales sino de todos los de su tipo como los peces de un cardumen pintados de lila, aunque hay un polígono también este no identifica individuos en el proyecto nos enfocaremos en la segmentación de instancias y la detección de objetos por como son una evolución un poco más natural de la clasificación.



¹la detección en un método más tradicional, sin embargo más robusto cuando no hay suficientes datos, al menos de que las áreas que ocupan los objetos sean de suma importancia es el método sugerido

Figura 1: diversas medidas de un modelo de clasificación



4.8. Medidas de rendimiento

Dado cierto modelo que quiere por ejemplo reconocer objetos del mundo real, podemos medir su rendimiento de diferentes formas, la forma más básica esta relacionada a dos medidas llamadas *exactitud* y *precisión*

Empezando por **exactitud**, esta básicamente nos dice que porcentaje de lo clasificado por el modelo es correcto, o de todos los ejemplos en que porcentaje acertó nuestro modelo, aunque esta medida es intuitiva, hoy día esta algo revaluada, sobre todo en problemas de clasificación donde las clases o bien no son iguales en proporción o bien condicionan implicaciones muy importantes (generalmente intervenciones medidas muy intrusivas) el primer problema es conocido como *Accuracy paradox* debido a esto nos tornamos a otras medidas como la especificidad, la sensitividad y el valor positivo predictivo estas dos ultimas conocidas en ML como *precisión* y *recall* son usadas por ejemplo una medida popular denotada F_1 que se calcula de la siguiente forma

$$F_1 = \frac{\text{PPV} \cdot \text{Sensitivity}}{\text{PPV} + \text{Sensitivity}} = \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

el poder de estas medidas es que son dos espectro de balancear una detección temprana (*recall*) con tener detecciones fiables(*precisión*), pues siempre podemos tener detecciones tempranas si marcamos

todo como una enfermedad o alerta, mientras que siempre podemos tener detecciones fiables si nunca nos arriesgamos a calificar nada como la enfermedad. por completitud las formulas de las anteriores medidas son las siguientes

$$A = \frac{TP + TN}{P + N} \text{ y } P = \frac{TP}{TP + FP}$$

El equilibrio entre precisión y exactitud debe ser considerado en situaciones análogas a estas: cuando no queremos tomar decisiones incorrectas e irreversibles (una amputación por ejemplo) es mejor priorizar la precisión, por otro lado si el costo de intervenir es bajo, como simplemente solicitar otra prueba no intrusiva o si por ejemplo no queremos dejar pasar el tiempo de algo grave (como un cancer) aumentar el recall puede ser importante

Este tipo de sacrificios permean los modelos de estadística y machine learning, y aunque en efecto en el laboratorio se puede llegar a tener estos valores en 100 % en el mundo real, donde hay ruido tanto en las mediciones como en los sistemas dinámicos lo normal es que debamos tomar una decisión sobre que priorizar y de acá medidas como la F_1 con pesos nos pueden permitir elegir algo más acorde a nuestra visión del problema ²

4.8.1. Intersection over Union

Aparte de estas medidas básicas para un modelo existen otras medidas dependiendo el área al que se aplican los mismos, en este caso para el análisis de imagen existe una medida llamada Intersection over Union o IoU, la forma de calcular la misma es simplemente con la división

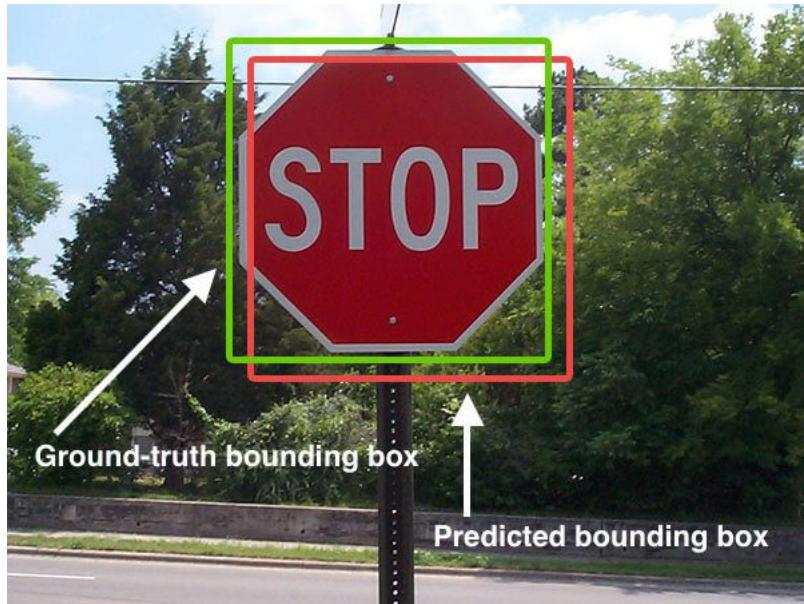
$$\frac{A \cap B}{A \cup B}$$

donde A y B son la áreas de el *ground truth* (o la aproximación poligonal de esta en los datos) y el área predicha por el modelo.

²véase por ejemplo el problema del Biass-Variance en la regresión

4.8.2. Regularization

4.8.3. Leakage



4.9. Aumentación y Limpieza de datos

Al momento de usar una red convolucional, a veces es importante la estandarización, bien sea de formatos de archivo, orientación o tamaño, si bien la lectura de diversos formatos es manejado generalmente por las librerías como *scikit learn* o *pandas*, para personas de areas como la agronomía puede resultar más comodo el uso de una interfaz gráfica, del mismo modo la aumentación de los datos es la idea de generar casos de entrenamiento sintetico como versiones rotadas de una imagen, con destellos o borrosas, pues este tipo de situaciones se daran en campo y es importante que el modelo pueda lidiar con ellas, por eso en la sección siguientes de Materiales y métodos pretendemos mostrar el paso a paso de este procedimiento sobre nuestro data set, con la ventaja de que al ser realizado en *roboflow*, una herramienta moderna web, casi cualquier persona puede replicar el proceso.

5. Materiales y métodos

5.1. Keras y Pytorch

La primera implementación de un modelo para el proyecto estaba en una red neuronal en Keras interfaz amigable sobre Tensorflow aunque la misma tiene una acogida importante en el análisis de datos, la

busqueda de la literatura nos llevo a considerar su competencia más importante *Pytorch*, esta tiene una filosofía diferente del manejo de las redes neuronales, las siguientes dos librerías para segmentación estan pues implementadas en esta ultima por lo que se debió migrar los conocimientos a esta nueva herramienta.

5.2. Detectron2 y YOLOv5

Con el fin de mejorar el rendimiento del modelo de clasificación, se realiza la segmentación de las imágenes para obtener la detección de hojas de aguacate. Uno de los primeros acercamientos fue haciendo uso de la librería Detectron2 de Facebook. El proceso consiste en tres pasos, etiquetado, entrenamiento y validación.

Para realizar el etiquetado de las hojas, se utiliza la herramienta labelme, el cual consiste en marcar el contorno de cada hoja en forma poligonal y asignar su etiqueta. Una ilustración del programa labelme se observa en la figura 2.

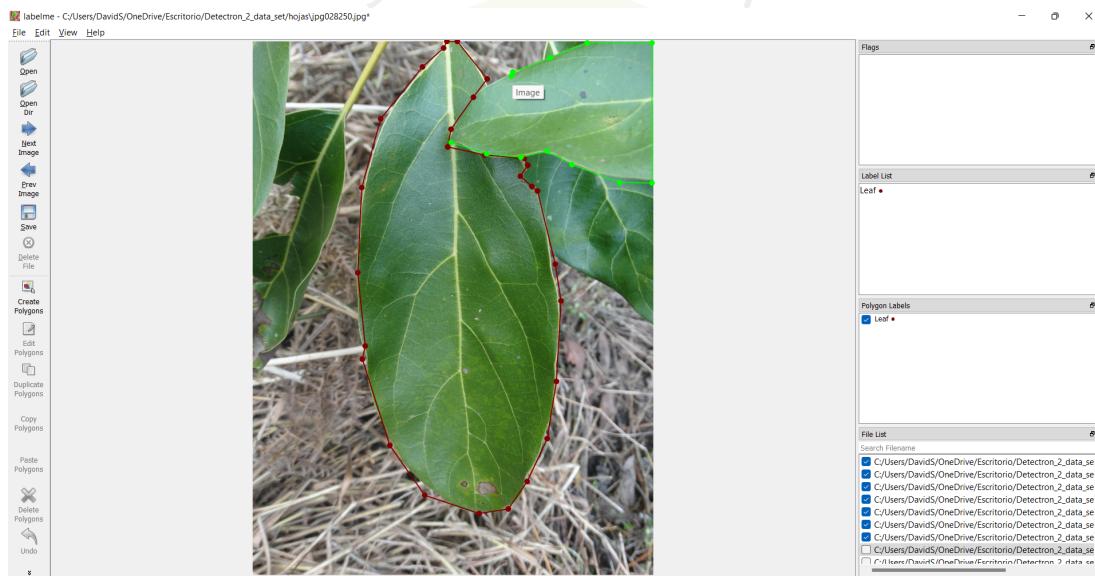


Figura 2: etiquetado en forma poligonal de las hojas usando labelme

Posteriormente, con el conjunto de datos etiquetados se entrena el algoritmo *Instance Segmentation*. En la figura 3 se muestra la predicción que realiza el algoritmo en 4 imágenes; en cada detección de una hoja, se obtiene una medida de certeza de que tanto se parece a una hoja.

Finalmente se realiza la validación con la medida (IOU), la cuál por el momento está pendiente.

otra alternativa considerada fue *YOLOv5* esta tiene documentación en cierta manera más amigable integración con múltiples herramientas como Weights and Bias para hacer análisis de la efectividad del entrenamiento o la próxima a explicarse Roboflow, *YOLOv5* además permite exportar los modelos a

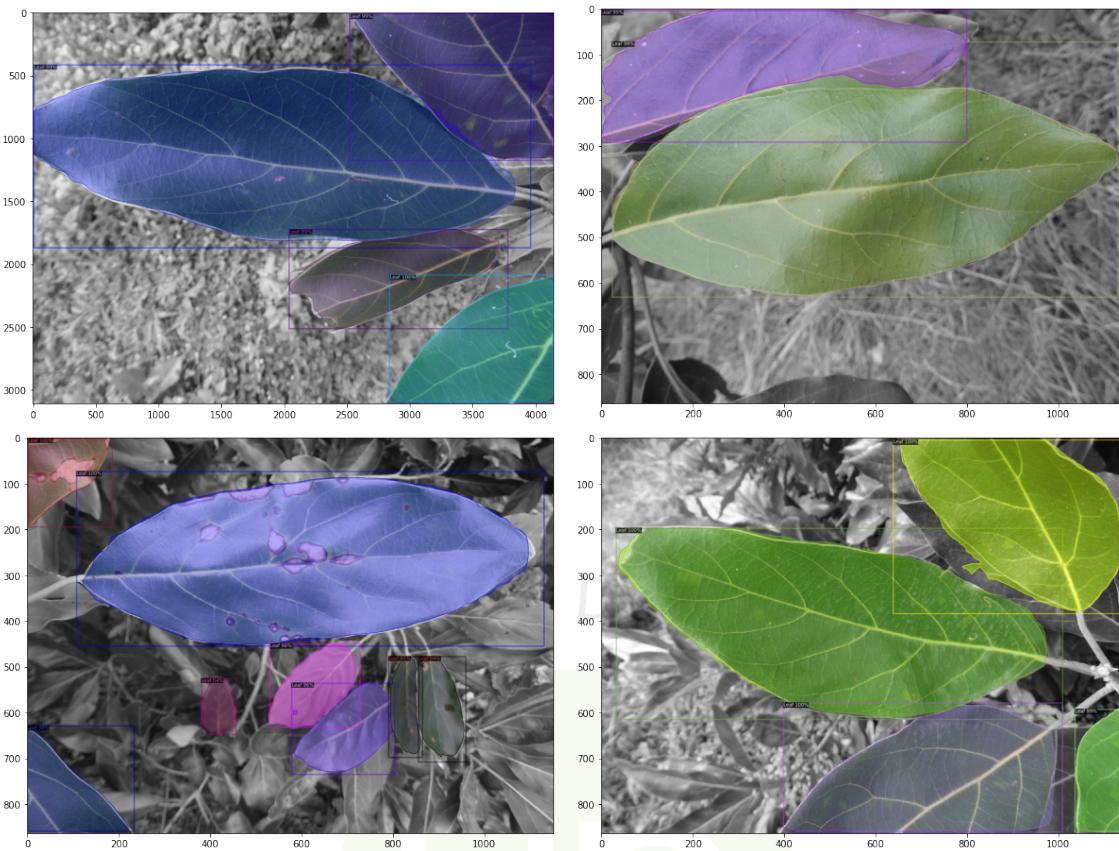


Figura 3: Predicción de hojas de Aguacate

diferentes interfaces de redes neuronales, permitiendo por ejemplo combinarla con una red en Tensorflow si fuera apropiado

5.3. Roboflow

Una de las mayores ventajas de conocer YOLOv5 fue llegar a Roboflow, Roboflow es un framework de desarrollo de visión por computadora con el que se puede realizar recopilación de datos e implementar técnicas de entrenamiento de modelos.

Al igual que labelme se puede realizar etiquetado de objetos, sin embargo, una ventaja que tiene sobre este es que modelos en si pueden asistir la detección de objetos, facilitando el etiquetado, también tiene un control de certeza del modelo usado, el cual permite detectar más objetos con la ventaja de que es más fácil eliminar detecciones o recuadros incorrectos que la creación de estos mismos

gracias a Roboflow nos fuimos por un modelo de *semisupervised learning*

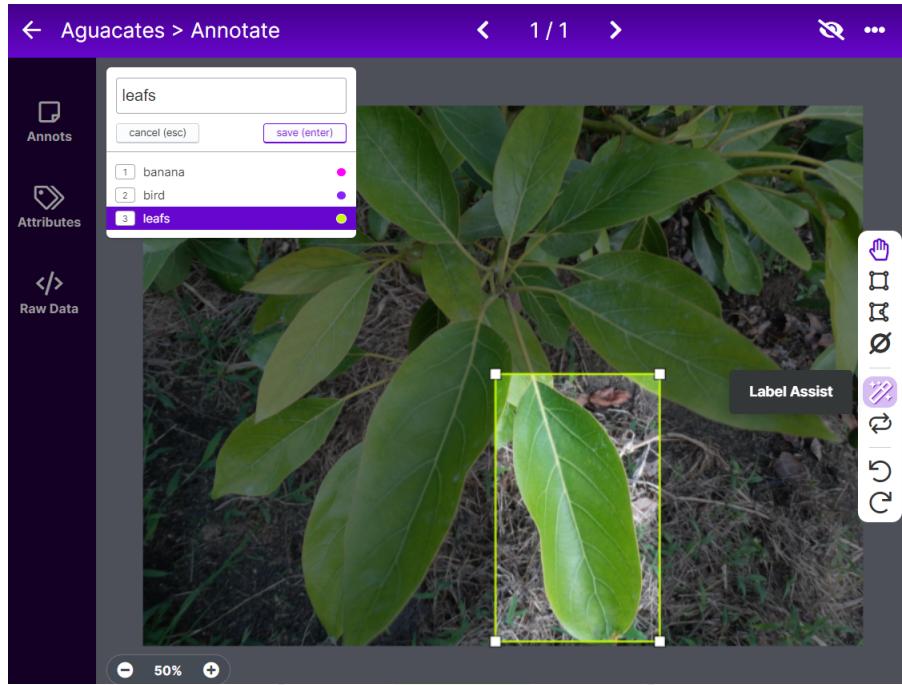


Figura 4: etiquetado de hojas en forma de caja usando Roboflow

6. Semi supervised

El aprendizaje supervisado, generalmente es hecho sobre datasets clásicos como los de los pasajeros del Titanic o las imágenes de *COCO* que muchas personas usan como aprendizaje o medida de la calidad de una red específica, pero que no tienen tanta relevancia en la actualidad o bien son muy generales para una aplicación práctica, se requieren más datos específicos del problema, estos son posibles en industrias de altos ingresos (entretenimientos, internet, defensa) pues son ellas quienes pueden aceptar el gran costo de no solo pagar a personal para clasificar ciertos objetos, tareas más difíciles como separar una persona de su fondo requiere ciertas habilidades y dedicación

Plataformas como

- Scale ai
- Sage maker ground truth

Cobran mensualidades o comisiones por cada dato a clasificar, se apoyan tanto de machine learning como de humanos encargados de clasificar los objetos que enviamos, por otro lado herramientas como *labelbox* o *roboflow* facilitan el proceso al permitir un trabajo en equipo más fluido o el uso de modelos para asistir en el etiquetado, aunque estas técnicas pueden ser logradas con software abierto, la flexibilidad y accesible de una plataforma web hacen muy atractivas estas técnicas, más aún la creación de pequeños modelos que luego usamos para continuar categorizando los datos de manera más rápida hacen parte de metodología semi supervisada.

7. Discusión

Haciendo uso del paquete labelme el proceso de etiquetado conlleva bastante tiempo ya que se hace manualmente, se considera que este procedimiento se puede realizar de forma eficiente si alternamos el uso de la librería Detectron2 y la aplicación Roboflow. Mejorando la forma en la que se realiza el etiquetado podemos aumentar la precisión del aprendizaje de las redes neuronales.

Los ensayos realizados están enfocados en la detección de hojas para obtener una familiarización con las librerías y el sistema de etiquetado, el siguiente paso es hacer el entrenamiento en si y comparar el modelo al menos con el anterior de Tensorflow, y si es posible comparar entre los framework *Detectron2* y *Yolov5*.

Agradecemos mucho la atención prestada

Referencias

- [1] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2.” <https://github.com/facebookresearch/detectron2>, 2019.