

UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

TALLER SISTEMAS CIBER-FÍSICOS

Detección y clasificación de malezas en cultivos

2024



**FACULTAD DE
INGENIERÍA
UDELAR**

INTEGRANTES

EDISON ESTRAMIL 5.121.926-4
MIGUELÁNGEL DÍAZ 5.214.428-4
DANIEL PADRON - 5.147.163-4

DOCENTES

Mercedes Marzoa
Gonzalo Tejera

INTRODUCCIÓN	3
Motivación	3
El problema y sus objetivos	3
Punto de partida y primeras limitaciones	4
INVESTIGACIÓN DE TÉCNICAS Y MODELOS	5
Elección del modelo	5
Técnicas de LSOD - Transferencia de conocimiento	6
GENERACIÓN DEL CONJUNTO DE DATOS	7
Obtención de instancias y conformación del conjunto de datos.	7
Roboflow	8
Aumento de datos	9
Soporte de Yolo y Roboflow para Aumento de datos.	9
Redes generativas antagónicas	10
ENTRENAMIENTO DEL MODELO	12
Infraestructura de Entrenamiento	12
Aplicación de Transferencia de aprendizaje	13
Pre entrenamiento con el conjunto de datos globales de cabezas d trigo de 2020 (GlobalWheat2020)	14
Optimización Hiperparámetros	15
Entrenamiento del modelo definitivo	19
EVALUACIÓN DEL MODELO	21
Modelo final	21
Métricas	21
Tiempo de inferencia	23
CONCLUSIONES	24
TRABAJO FUTURO	26
GLOSARIO	27
BIBLIOGRAFÍA	28

INTRODUCCIÓN

En el siguiente documento se presentará el desarrollo y resultados del proyecto de “Reconocimiento de Malezas” desarrollado en el contexto del curso de Taller de Sistema Ciber-Físicos en su edición correspondiente al primer semestre del año 2024.

En esta primera sección introduciremos al lector al contexto en que se enmarca el proyecto, que lo motiva, sus objetivos y finalmente se describe la organización del resto del documento.

Motivación

Se conoce como Maleza a cualquier tipo de planta no querida que se encuentra compitiendo por recursos con las que sí se buscan cultivar. Se ha observado que tener un manejo “pobre” en lo que respecta a la eliminación de malezas ha causado pérdidas de hasta un 50% en el cultivo, y en el caso de que el manejo sea nulo, en otras palabras, las malezas estén sin control, se han reportado pérdidas de hasta un 90% (Dang et al.). Por lo tanto, se ha propuesto un ambicioso proyecto de control de malezas vía un robot.

Este último tiene como propósito ser capaz de recorrer los cultivos, reconocer las malezas, y eliminarlas - idealmente - de forma que no implique el uso de pesticidas. El uso de estos, es caro, perjudicial en el ámbito medioambiental y en lo que respecta a la salud de quienes consumen alimentos que han sido expuestos a dichos agroquímicos así como de las personas que lo aplican. Además, puede llegar a provocar una pérdida en la calidad del cultivo, y especialmente aumenta la resistencia de las malezas que se busca remover.

En un esfuerzo por colaborar con el desarrollo del proyecto antes mencionado, se comenzó a realizar los primeros pasos en lo que respecta al sistema de reconocimiento de malezas.

En consecuencia, el proyecto que describiremos en este documento se enmarca como uno de los primeros intentos en desarrollar el componente de reconocimiento de malezas para el robot antes mencionado.

El objetivo es la realización de un sistema capaz de ubicar las diferentes malezas buscadas dentro de una imagen dada, discerniendo entre las diferentes especies. Idealmente, el sistema de reconocimiento podría recibir del robot las imágenes capturadas por sus cámaras, y posteriormente el sistema de reconocimiento respondería con las ubicaciones y especies de las malezas reconocidas en la imagen; lo que permitiría al robot tomar acciones sobre las malezas encontradas.

El problema y sus objetivos

El problema abordado se puede sintetizar como “Dada una imagen que puede tener o no malezas, reconocer la ubicación y especie de las mismas”.

Como se puede observar, la descripción del problema no es exhaustiva, por ejemplo, no se especifica que tipos de malezas se buscan reconocer, en qué ángulo con respecto al suelo serán tomadas las imágenes, cuál se espera que sea el “Background” de las imágenes o en qué estación del año se espera que el reconocimiento sea exitoso.

El motivo por cuál la respuesta a estas preguntas no integran la especificación del problema es que aún no se encuentran definidas debido a que el desarrollo del robot recién se encuentra en sus etapas iniciales. Se ha decidido entonces no imponer estas restricciones al desarrollo del componente de reconocimiento en su etapa inicial.

Además, por limitaciones en lo que respecta al conjunto de datos - que luego mencionaremos - se decidió únicamente trabajar sobre tres especies de malezas: *Lolium*, *Cyperus Rotundus* e *Ipomoea*.

También por el mismo motivo no asumimos ninguna época específica del año o crecimiento de la planta. No se han impuesto restricciones en lo que respecta a la metodología en cómo han sido capturadas las fotografías utilizadas en el proceso de entrenamiento del modelo, en particular no se han impuesto restricciones en lo que respecta al ángulo de captura o background. Por lo tanto se propone como objetivo entrenar un modelo de aprendizaje automático capaz de detectar “de la mejor manera posible” - más adelante desarrollaremos cómo cuantificar ese objetivo - malezas en una imagen discerniendo su especie.

Punto de partida y primeras limitaciones

El problema que buscamos resolver se lo conoce como detección de objetos (Object Detection en inglés), él mismo se encuentra dentro del área de visión por computadora. En esencia, este tipo de problemas se enfoca en detectar varios objetos dentro de una imagen, encuadrando el objeto detectado, marcando la clase a la que corresponde y la confianza que tiene el modelo en que la predicción es correcta. Luego, en la sección [Investigaciones y modelos encontrados](#) se explorará más acerca del estado del arte en las arquitecturas de aprendizaje automático utilizadas para este tipo de problema.

Es importante resaltar que el problema a resolver es de extrema relevancia para el área de visión por computadora, lo que es una gran señal, ya que significa que se cuenta con una gran cantidad de material y herramientas de soporte para este proyecto.

Una gran primera limitación es que no se contaba con un conjunto de datos, lo que podría impedir cualquier intento de entrenamiento de un modelo. En la sección [Generación de conjunto de datos](#) se profundizará sobre cómo se abordó esa carencia.

En la siguiente sección presentaremos el estado del arte en cuanto a los modelos y algunas de las técnicas más utilizadas para el problema de detección de objetos por computadora, se introducirá al lector en como fue tratado y generado el conjunto de datos, posteriormente veremos el proceso de entrenamiento y la evaluación del modelo, finalizando con las conclusiones y posible trabajo futuro.

INVESTIGACIÓN DE TÉCNICAS Y MODELOS

Elección del modelo

Existen múltiples modelos de aprendizaje automático para el problema de detección de objetos, aunque actualmente, la enorme mayoría de ellos, utilizan técnicas de aprendizaje profundo vía redes neuronales. Actualmente, hay dos familias de estos modelos: los de dos etapas y los de una etapa.

Dentro de los modelos de dos etapas, los más conocidos son los de las familias de las R-CNN - Red convolucional de regiones (Regions with Convolutional Neural Network). Dentro de esta familia se encuentran tres generaciones, el modelo R-CNN, Fast R-CNN - R-CNN Rápida (Fast Regions with Convolutional Neural Network) y Faster R-CNN - R-CNN Más Rápida (Faster Regions with Convolutional Neural Network). La primera etapa propone una serie de regiones donde es posible que se encuentre un objeto, mientras la segunda toma cada una de esas regiones, la clasifica y hace una regresión de donde poner los rectángulos que encuadran a los objetos encontrados. La primera etapa puede ser realizada de muchas formas, la primera generación de las R-CNN utilizaba el algoritmo búsqueda selectiva, mientras que actualmente las Faster R-CNN utilizan redes neuronales para este propósito, conocido como redes de propuestas de regiones.

Una de las principales ventajas de esa familia de redes, en especial las Faster R-CNN, es que actualmente son uno de los métodos que consiguen mejores predicciones. Observar que la etapa de propuestas de regiones sugiere un gran número de regiones, las cuales deberán ser luego procesadas por la segunda etapa (clasificación y regresión), lo que aumenta significativamente los tiempos de inferencia, ya que una misma parte de una imagen es procesada varias veces. Además, esta arquitectura no tiene un único componente, sino que el pipeline está conformado por una serie de redes que se deben entrenar, lo que produce que el tiempo de entrenamiento sea mayor con respecto a los modelos de una etapa.

Por otro lado, los modelos de una etapa, se caracterizan por encapsular la totalidad de la arquitectura (detección, clasificación y regresión) en una misma red convolucional. Una de las arquitecturas más conocidas con esta característica es YOLO - Solo miras una vez (You only look once). YOLO es una arquitectura de código abierto, diseñada específicamente para el problema de detección de objetos, fue diseñada con el fin de ser utilizada en tiempo real, por lo que sacando partido de su diseño en una única etapa, logra tiempos de inferencia y de entrenamiento sustancialmente inferiores a las arquitecturas de la familia R-CNN.

YOLO además de ser uno de los modelos más populares, también se encuentra en una etapa madura en su desarrollo. Al momento de iniciar el proyecto su décima versión ya está disponible junto con su implementación. Esto permite entrenar y evaluar el modelo sin necesidad de implementar desde cero la red neuronal, ahorrando tiempos de implementación, los cuales han sido destinados a probar diferentes técnicas y configuraciones que se irán mencionando a lo largo de este documento.

Dentro de YOLOv9 se tienen múltiples modelos: YOLOv9t, YOLOv9s, YOLOv9m, YOLOv9c y YOLOv9e. Estos modelos difieren en el número de parámetros, donde YOLOv9t es el que tiene menor número de parámetros (2 millones) y YOLOv9e es el que tiene la mayor cantidad (58,1 millones). La elección del modelo es un "trade-off" entre el valor de la mAP (ver [GLOSARIO](#)) y el tiempo de inferencia (y entrenamiento): mientras menos parámetros tiene la red, más rápida es, pero obtiene peores resultados en términos de la mAP.

Este es un proyecto que necesita un alto rendimiento, por lo que no parece correcto elegir uno de los modelos más simples. Sin embargo, con el fin de probar diferentes técnicas, y por ende, realizar múltiples pruebas, es deseable minimizar el tiempo de entrenamiento. Por todo lo anterior, consideramos que utilizar el modelo YOLOv9c es la mejor opción, ya que es recomendado por el equipo de Ultralytics debido a su equilibrio entre rendimiento y eficiencia.

Técnicas de LSOD - Transferencia de conocimiento

Dentro del campo de investigación de Detección de Objetos existe una área llamada LSOD - Detección de objetos con pocas muestras (Low-Shot Object Detection), la cual tiene como motivación estudiar y buscar técnicas para mejorar el desempeño de las redes que han sido entrenadas con pocas instancias, en particular el área FSOD - Detección de Objetos con Pocos Ejemplos (Few-Shot Object detection) , OSOD - Detección de Objetos con un Solo Ejemplo (One-Shot Object detection) y ZSOD - Detección de Objetos sin Ejemplos (Zero-Shot Object Detection). Debido a que no se contaba con un conjunto de datos en el comienzo del proyecto, y las capacidades para formar uno son limitadas, se procedió durante el desarrollo estudiar técnicas en el área de FSOD con el fin de obtener técnicas que pudieran mejorar el desempeño.

Las dos técnicas más utilizadas son conocidas como Meta-Aprendizaje(Meta Learning) y Transferencia-Aprendizaje(Transfer Learning). Meta-Aprendizaje consiste en utilizar un conjunto de datos base (gran cantidad de instancias, como el conjunto de datos de COCO), y durante su entrenamiento la red no solo aprende de sus instancias, sino que también desarrolla la capacidad de aprender eficazmente de nuevas tareas a través del entrenamiento en un conjunto de tareas. Esto permite repetir el entrenamiento con un conjunto de datos con pocas instancias, la red pueda adaptarse y aprender de manera más eficiente, mejorando los resultados con un número reducido de ejemplos. Esto permite que a la hora de repetir el entrenamiento en el conjunto con pocas instancias logre mejores resultados. Por otro lado, Transferencia-Aprendizaje consiste en “transferir” el conocimiento aprendido por un modelo a otro. Anteriormente, las técnicas que empleaban Meta-Aprendizaje obtenían mejores resultados que las de Transferencia-Aprendizaje, hasta que se comenzó con una metodología llamada TFA - Enfoque de Afinado Fino en Dos Etapas (Two-Stage Fine-tuning Approach), donde a partir de ese momento fueron las técnicas de Transferencia-Aprendizaje las que logran una mayor mejora (HUANG et al.).

La técnica TFA consiste en entrenar el modelo con el conjunto de datos base, permitiendo que todas las capas se actualicen durante la Retropropagación (Backpropagation), luego de haber finalizado se debe comenzar nuevamente con el entrenamiento, pero esta vez

utilizando el conjunto de datos nuevo, utilizando los pesos con los que finalizó el anterior proceso, sin permitir actualizar las capas convolucionales de la red durante este segundo entrenamiento. Esto fuerza a que el extractor de características esté dado por lo que se aprende del conjunto de datos base, y luego sean las capas de clasificación y la de regresión las que se adapten al nuevo conjunto de datos.

En síntesis: Se realiza el entrenamiento dos veces, primero con el conjunto de datos base y luego con el nuevo conjunto de datos original, pero en el segundo entrenamiento utilizando el estado con el que finalizó el anterior entrenamiento, y no se permite actualizar las capas convolucionales.

En la sección [ENTRENAMIENTO DEL MODELO](#) procederemos a desarrollar como se puede llevar esta técnica a la implementación de YOLO suministrada por Ultralytics.

GENERACIÓN DEL CONJUNTO DE DATOS

Obtención de instancias y conformación del conjunto de datos.

La conformación del dataset fue una de las tareas más complicadas debido a la falta de imágenes etiquetadas de las malezas de interés. Realizamos una búsqueda exhaustiva, pero no fue posible encontrar plantas similares, dada la falta de datos en general, la gran variedad de plantas y su dependencia regional.

La única alternativa disponible era crear un conjunto de datos desde cero. Esta tarea no es sencilla, ya que no solo implica etiquetar cada ubicación de la planta dentro de la imagen, sino que también requiere un análisis profundo para identificar las mejores imágenes que aporten valor al entrenamiento. Las imágenes se seleccionan de forma muy meticulosa, asegurando ciertos ángulos, capturando la planta en diferentes etapas de crecimiento y épocas del año para obtener una representación completa y mejorar la precisión de las predicciones.

Un ejemplo de esto se encuentra en el artículo (Wang P, and Tang Y, #) , donde se describe una metodología para una especie que consiste en capturar imágenes de la planta en etapas de una a cinco hojas. En el mismo estudio se menciona que las imágenes son evaluadas y aceptadas o rechazadas por hasta tres expertos.

La primera tarea fue conseguir las imágenes sin etiquetar. Para ello, utilizamos la plataforma GBIF (GBIF), que recopila información de plantas en el mundo, incluyendo una cantidad considerable de imágenes para cada especie. El problema con estas imágenes es que fueron tomadas para notificar la ocurrencia de la planta y no con fines de construcción de un conjunto de datos. Por lo tanto, encontramos una amplia variedad de imágenes no útiles.

Se utilizó un criterio para seleccionar únicamente las imágenes que podían ser representativas del problema, aquellas que se veían de frente y con mejor calidad. Sin embargo, no se pudo realizar un proceso de transición entre etapas de crecimiento de la planta ni capturar otras facetas debido a las estaciones climáticas.

Roboflow

Para la creación y mantenimiento del conjunto de datos se utiliza una herramienta llamada Roboflow, diseñada específicamente para la creación y distribución de conjuntos de entrenamiento para aprendizaje por computadora.

Roboflow permite agilizar la creación de conjunto de datos personalizados gracias a su interfaz ágil y dinámica para realizar el etiquetado de cada imagen. Además, cuenta con una versión de autoetiquetado que, a través de una petición, realiza de forma automática los rectángulos en la imagen (bounding boxes). Esta función fue especialmente útil para la identificación de la planta *Ipomoea*, ya que la petición era sencilla y solo se debía pedir que buscara elementos similares a flores.

Sin embargo, para el caso de *Cyperus* y *Lolium*, la herramienta no fue tan efectiva. Debido a la complejidad de estas plantas, no fue posible describir un patrón o forma específica para que el autoetiquetado funcionara correctamente. En la página de Roboflow, se recomienda utilizar la petición de una manera sencilla, similar a cómo se le explicaría a un niño de diez años. Esto significa que podemos describir ciertos patrones, pero no de manera muy compleja.



Figura 1 - Imagenes etiquetadas del dataset confeccionado usando la herramienta Roboflow.

Como resultado, se realiza el etiquetado de *Cyperus* y *Lolium* de forma manual. Este proceso fue laborioso, pero permitió asegurar la precisión en la identificación y etiquetado de estas plantas. En Figura 1 se encuentran imágenes de *Cyperus*, *Ipomoea* y *Lolium* respectivamente, aquí se puede observar que la imagen correspondiente a *Lolium* requirió una gran cantidad de cuadrados delimitadores realizados de manera manual.

Se logró realizar un total de 300, 393, 426 imágenes etiquetadas de *Ipomoea*, *Cyperus* y *Lolium* respectivamente.

Las imágenes al ser recuperadas de una página que no es específicamente para generación de conjunto de datos, tienen resoluciones variadas, este punto aunque afecta al aprendizaje no es un problema porque YOLO reasigna la resolución de cada imagen en base a un parámetro dado, para este caso se usó resolución de 640x640.

Esto también genera problemas en el aprendizaje de los patrones, si bien se intentó recopilar las imágenes siguiendo un criterio como ser aquellas imágenes que se acerquen más al ambiente donde operaría el robot, estas imágenes no llegaban a abarcar la información necesaria para realizar un entrenamiento exhaustivo, como se mencionó anteriormente faltaron aspectos como las etapas de crecimiento o imágenes de la planta en diferentes estaciones del año.

Una ventaja adicional de utilizar Roboflow es su capacidad para gestionar y mejorar los datasets a lo largo del tiempo. La herramienta permite realizar ajustes y mejoras continuas en el etiquetado, facilitando la actualización del dataset conforme se obtienen nuevas imágenes o se identifican errores en el etiquetado inicial.

En resumen, la utilización de Roboflow no solo agilizó el proceso de etiquetado, sino que también proporcionó herramientas para mejorar y mantener la calidad del dataset.

Aumento de datos

Aumento de datos (Data Augmentation) es una técnica utilizada en aprendizaje por computadoras y visión por computadoras para aumentar la cantidad y diversidad de datos disponibles para entrenar los modelos. Esto se logra aplicando transformaciones a las imágenes existentes, como rotaciones, traslaciones, escalados, volteos, cambios de brillo, y más. El objetivo es mejorar la capacidad del modelo para generalizar y reconocer patrones, al exponerlo a una mayor variedad de ejemplos sin necesidad de recolectar nuevas imágenes. Esta técnica es especialmente útil cuando se dispone de un conjunto de datos limitado, ya que permite crear un conjunto de datos más robusto y diverso a partir de las imágenes originales.

Soporte de Yolo y Roboflow para Aumento de datos.

Tanto YOLO como Roboflow ofrecen herramientas integradas para realizar el aumento de datos.

Inicialmente, se utilizan las capacidades de aumento de Roboflow, aplicando rotaciones de 15 y -15 grados a las imágenes. Se opta por este tipo de aumento específico porque cada transformación adicional incrementa el tiempo de entrenamiento, y no se quería sobrecargar el proceso en el primer acercamiento con YOLO.

Sin embargo, al investigar más a fondo, se descubrió que YOLO ya realiza varias técnicas de aumento de datos de manera predeterminada. Estas incluyen:

- **Escalado (Scaling):** Ajusta el tamaño de las imágenes.
- **Desplazamiento (Translation):** Mueve las imágenes horizontal y verticalmente.
- **Rotación (Rotation):** Gira las imágenes en varios grados.
- **Volteo (Flipping):** Invierte las imágenes horizontalmente.

- **Cambio de brillo (Brightness Adjustment):** Modifica el brillo de las imágenes.
- **Cambio de saturación y contraste (Saturation and Contrast Adjustment):** Ajusta la saturación y el contraste.

Estas transformaciones se realizan variando parámetros entre valores iniciales y finales, asegurando una amplia diversidad en los datos de entrenamiento. Se decide eliminar las rotaciones aplicadas con Roboflow, ya que eran redundantes y no aportaban valor adicional a los aumentos ya implementados por YOLO.

Redes generativas antagónicas

Las redes GAN - Red Generativa Adversarial (Generative Adversarial Network) son una arquitectura definida por dos funciones (**Generador y Discriminador**), las cuales suelen ser implementadas con redes neuronales que compiten en un juego que, luego de varias iteraciones de entrenamiento, tiene como resultado dos cajas negras que llevan a cabo las tareas de **generar** imágenes similares a un dataset dado, y **discriminar** entre imágenes reales y falsas. En algunos estudios en el área de medicina, (Hammami et al.) se utilizan las GAN en problemas de detección y clasificación, consiguiendo mejoras con los modelos YOLO de versiones previas a la de este trabajo, por lo que se considera un posible buen camino para seguir. Más adelante en el informe se aplicarán las técnicas para comprobar si significa una mejora también en esta realidad. Esta técnica se puede aplicar de dos formas distintas:

- ❖ Generación de imágenes desde cero con DCGAN - Red Generativa Adversarial Convolucional Profunda (Deep Convolutional Generative Adversarial Network)
 - Las DCGAN son redes ampliamente usadas en generación de imágenes debido a su capacidad para generar contenido visualmente realista. Son en grandes rasgos, la arquitectura GAN pero con las funciones Generador y Discriminador implementadas mediante redes CNN. En primer lugar se lleva a cabo un proyecto “hola mundo” para determinar el costo de entrenar una red antagónica con datos obtenidos del dataset FASHION-MNIST y evaluar si es posible, dado el contexto, generar más imágenes con este enfoque. Debido al límite de recursos y el problema de etiquetar dichas imágenes generadas, se opta por finalizar esta línea de investigación y optar por el aplicado de filtros. Sin embargo, se señala como posible método de transferencia de aprendizaje mediante la combinación con otros modelos de detección y etiquetado.
- ❖ Aplicado de filtros a imágenes ya existentes con CycleGAN - Red Generativa Adversarial Consistente con Ciclos (Cycle-Consistent Generative Adversarial Network)
 - El trabajo citado (Hammami et al.) utiliza este tipo de redes que permiten la transformación de imágenes, se encuentran disponibles algunos modelos de esta red para aplicar filtros a las imágenes del dataset lo que conlleva a evadir el problema de la falta de recursos al utilizar modelos pre entrenados y además no posee el problema de etiquetar estas instancias, ya que, al

funcionar como un aplicado de filtros, la caja sería la misma que en la imagen original. En la Figura 2 se puede observar un ejemplo de generación.

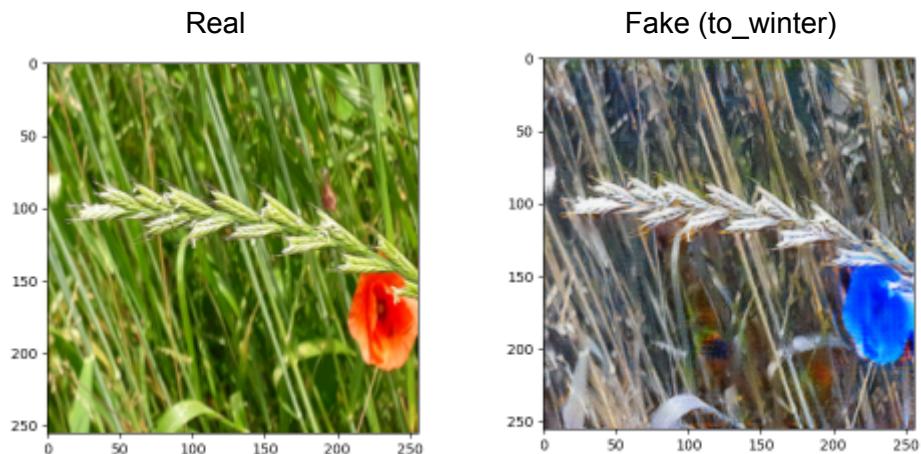


Figura 2 - aplicación de una red CycleGAN sobre una instancia de *Lolium Multiflorum* perteneciente al dataset de entrenamiento.

ENTRENAMIENTO DEL MODELO

Infraestructura de Entrenamiento

Para la infraestructura de entrenamiento, se utilizaron tres métodos distintos.

Se presenta una descripción de cada opción utilizada, así como sus ventajas y desventajas.

1. Tarjeta Gráfica Dedicada

Inicialmente, se utiliza una tarjeta gráfica dedicada para el entrenamiento. Esto nos permitió obtener los primeros resultados y familiarizarnos con el proceso.

- **Ventajas:**

- Acceso inmediato y control total sobre el equipo.
- No requiere aprendizaje adicional sobre nuevas plataformas.

- **Desventajas:**

- Dependencia de la disponibilidad del equipo, lo que limita su uso para otras tareas.
- Alto consumo de recursos, dejando la computadora exclusiva para el entrenamiento.

2. Google Colab

La segunda opción fue Google Colab, una plataforma ampliamente utilizada para entrenamiento de modelos de machine learning.

- **Ventajas:**

- Facilidad de uso y configuración rápida.
- Acceso a GPUs sin costo en la versión gratuita.
- Colaboración en tiempo real con otros miembros del equipo.

- **Desventajas:**

- Limitaciones en la versión gratuita, como la posibilidad de interrupción de la ejecución en cualquier momento.
- Riesgo de pérdida de progreso debido a interrupciones inesperadas.
- Restricciones de tiempo de uso continuo y recursos disponibles.

3. ClusterUy

Finalmente, se hizo uso de ClusterUy, un cluster de computación de alto rendimiento.

- **Ventajas:**

- Capacidad de procesamiento más robusta y recursos dedicados.
- Ejecución independiente entre los miembros del grupo.

- **Desventajas:**

- Curva de aprendizaje inicial para la utilización del cluster y la gestión de objetos.
- Problemas iniciales con la finalización inesperada de sesiones, lo que resultó en la pérdida de datos.

Cada método ofrece diferentes beneficios y desafíos. La elección de la infraestructura adecuada dependió de la etapa del proyecto y de las necesidades específicas del entrenamiento. Eventualmente, la combinación de estas opciones permitió optimizar el proceso de entrenamiento y mejorar la eficiencia.

Aplicación de Transferencia de aprendizaje

El modelo utilizado es YOLOv9c, el mismo cuenta con más de 25 millones de parámetros a optimizar, en consecuencia técnicas como Transferencia de aprendizaje son ampliamente utilizadas con el fin de aumentar la velocidad de aprendizaje. YOLO en su novena versión provee dos formas de aplicar estas técnicas: partiendo de un modelo pre entrenado, y congelando sus capas convolutivas.

Todos los modelos de YOLO puede ser obtenidos juntos con los pesos resultantes del entrenamiento utilizando el conjunto de datos COCO¹.

En consecuencia, se optó por el uso de los pesos pre entrenados en el resto de los entrenamientos ejecutados, especialmente por los siguientes dos motivos:

- Entrenar un modelo sin utilizar pesos pre entrenados es un proceso que requiere un mayor número de épocas en realizarse, sobre todo considerando el gran número de parámetros que las capas convolucionales contienen. Es de especial importancia obtener resultados significativos en el menor número de épocas posibles, sobre todo considerando los elevados costos del proceso de optimización de hiperparámetros, el mismo se profundiza en la sección [Optimización Hiperparámetros](#).
- Con el fin de optimizar la función de costo, es preferible partir de valores de pesos más cercanos al óptimo que una configuración aleatoria.

Durante el proceso de entrenamiento se puede optar por no permitir que determinadas capas actualicen sus pesos (se suele referir a esta acción como “congelar” una capa), según lo expuesto en la sección [Técnicas de LSOD - Transferencia de conocimiento](#), en particular a lo que respecta a la técnica TFA, hay casos donde aplicar tal congelamiento es beneficioso en términos de precisión. Notar que al ser menor el número de capas a actualizar, el tiempo de cómputo requerido por época es sustancialmente menor. En la Figura 3 se expone el rendimiento del modelo congelando sus primeras diez capas, y uno que no aplica ningún tipo de congelamiento. Las diez primeras capas son conocidas como el Esqueleto (Backbone), compuestas principalmente por capas convolucionales.

¹ COCO (En inglés Common Object in Context) es un conjunto de datos el cual contiene 80 categorías de los objetos “comunes”. Contiene más de 220 mil imágenes etiquetadas.

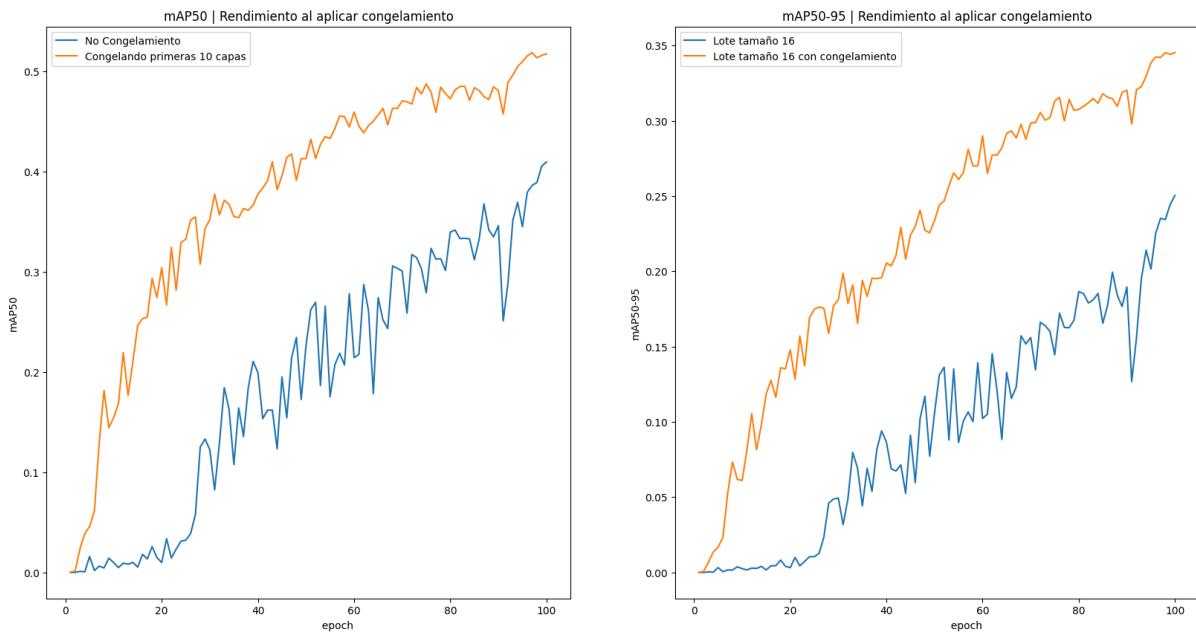


Figura 3 - Comparación de rendimiento en el uso de congelamiento

Observar que congelar las primeras capas ha sido beneficioso en cuanto al rendimiento, especialmente en lo que respecta en que tan rápido el modelo aumenta el valor de sus mAPs. Como profundizaremos en la sección [Entrenamiento del modelo definitivo](#), congelar las capas permite un aumento drástico en la velocidad de aprendizaje, como era esperado por lo relatado en la técnica TFA, pero no por eso obtiene mejores resultados luego de haber finalizado el entrenamiento comparado con modelos que no han sido entrenados con congelamiento. En el cómputo del modelo final se realizará el modelo entrenado tanto con congelamiento como sin él.

Pre entrenamiento con el conjunto de datos globales de espigas de trigo de 2020 (GlobalWheat2020)

GlobalWheat2020 (“Global Wheat”) es un conjunto de datos de espigas de trigo compuesto por alrededor de 4000 imágenes el cual fue utilizado en una competencia la cual consistió en encontrar un modelo de aprendizaje automático que mejor resuelva el problema de detectar cabezas de trigo, en esta competencia, algunos de los modelos utilizados fueron versiones previas de YOLO. Se considera esta opción para obtener pesos pre entrenados debido a que se trata de un enfoque más específico que el del dataset COCO y siendo las formas muy similares a la de Lolium. Partiendo de los pesos de COCO², tomamos los pesos después de un entrenamiento de 50 épocas en un modelo versión 9c de YOLO, debido al tiempo que insumió entrenar (aproximadamente 10 minutos por época) con este conjunto de datos se eligió utilizar 50 épocas.

Se procede a entrenar la mejor versión conseguida hasta el momento pero con la diferencia de que este entrenamiento partirá desde dichos pesos iniciales obtenidos mediante dicho conjunto de datos de imágenes de trigo. Previamente a esta prueba, se reformulan los etiquetados de las imágenes de Lolium, los cuales consisten en etiquetar la parte más

² En la pagina de ultralytics se sugiere partir desde los pesos de COCO
<https://docs.ultralytics.com/datasets/detect/globalwheat2020/>

característica de dicha maleza, es decir, la espiga que caracteriza en las imágenes del conjunto de datos, las cuales, son hasta cierto punto, similares a las del trigo. Como se puede observar en la Figura 4, se trata de un modelo similar que prácticamente no afecta el rendimiento de la clase Ipomoea, mientras que la clase Cyperus se ve afectada negativamente debido a que si bien aumenta la eficacia a la hora de no reconocer dicha maleza cuando en realidad es fondo (FP), el modelo pierde efectividad al medir los verdaderos cyperus, etiquetandolos como fondo (FN). Consideramos más importante este último punto ya que se trata del caso en donde no detectaríamos malezas que en realidad se deben quitar (FN) y eso es más importante que detectar algo que no lo es como una maleza (FP) debido a la realidad del problema.

Siguiendo con la lógica anterior, observamos una leve mejora en la detección de Lolium pero que no es significativa a nivel general del modelo por lo ya mencionado, por lo tanto, decidimos terminar con esta línea de investigación concluyendo que si bien no se descarta que pueda ser útil, solamente sería posible que lo fuese en un detector exclusivamente de malezas similares al trigo.

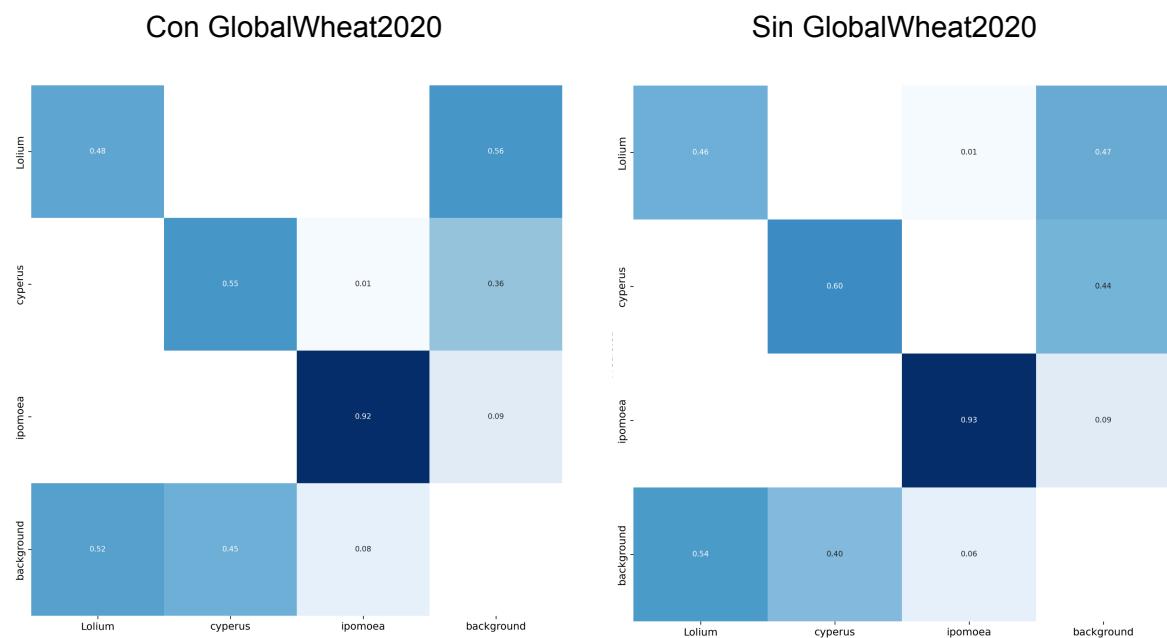


Figura 4 Matrices de confusión para la evaluación sobre los pesos del dataset global de cabezas de trigo, a la izquierda la matriz del modelo que utiliza estos pesos y a la derecha la que parte de los pesos de COCO. En el eje horizontal se indica la clase verdadera y en el vertical la predicha.

Optimización Hiperparámetros

Las redes neuronales, al igual que la gran mayoría de los métodos de aprendizaje automático, requieren de la determinación de una serie de hiperparámetros para posteriormente ser entrenadas. Encontrar el valor de los hiperparámetros que optimiza los resultados durante el entrenamiento es fundamental, ya que el rendimiento de los modelos es muy sensible a estos.

YOLO en su novena versión posee un gran número de hiperparámetros, algunos corresponden al optimizador (como por ejemplo la tasa de aprendizaje, o el momentum), mientras que otros corresponde al aumento de datos (como la aplicación de rotaciones, cambios en la saturación y brillo de las imágenes). Uno de los hiperparámetros más influyentes en el entrenamiento es la cantidad de instancias por lote (en inglés batch), sus valores posibles son: 1, 8 o 16 instancias. Puede ser incluso mayor, pero la infraestructura utilizada no tolera 32 o más instancias por lote. En la Figura 5 se presenta la diferencia en el rendimiento entre los modelos entrenados durante cien épocas, sin congelamiento y variando el número de instancias por lote entre 1, 8 y 16 instancias.

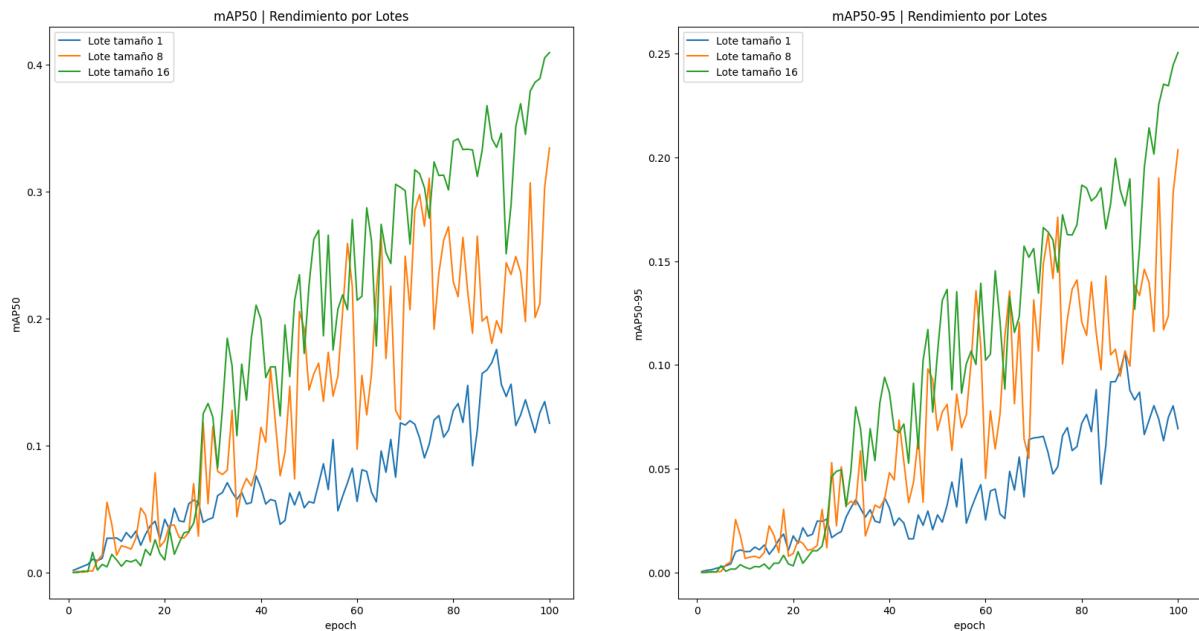


Figura 5 - Comparación en rendimiento utilizando lote de una y dieciséis instancias.

Se observa que el rendimiento es claramente superior cuando el número de instancias por lote es mayor. El resultado era esperable, ya que significa que previo a actualizar los pesos de la red, un mayor número de instancias son evaluadas, formando parte de la decisión del nuevo paso, pero sin ser demasiadas instancias, permitiendo así sobrellevar de mejor forma mínimos locales en la función de costo.

Aún restan la determinación de 28 hiperparámetros. Observar que probar de forma aleatoria combinaciones de estos parámetros raramente conduce a los valores óptimos. Con el fin de ofrecer un mejor mecanismo de búsqueda, Ultralytics público una herramienta que permite probar diferentes combinaciones de valor para los diferentes hiperparámetros, variando sus valores mediante algoritmos genéticos, en particular utilizando el método de Mutación³, lo que permite disminuir el número de entrenamientos.

Para evaluar que tan buena es una configuración de hiperparámetros se utilizó la siguiente función de costo: 10% del valor corresponde de la mAP con nivel de confianza 50%, mientras que el restante 90% corresponde el valor de la mAP con el rango de confianza 50% a 95%. En cada iteración el modelo es entrenado desde cero (aunque utilizando los

³ Para más información sobre el mecanismo de optimización de hiperparámetro ofrecido por YOLO puede consultar el siguiente enlace:

<https://docs.ultralytics.com/es/guides/hyperparameter-tuning/#genetic-evolution-and-mutation>

pesos preentrenados de COCO), ejecutando durante 30 épocas, y finalmente según los resultados obtenidos se proponen nuevos valores de hiperparámetros a ejecutar en la siguiente iteración. Se ejecutaron un total de 130 iteraciones.

El conjunto de datos utilizado para entrenar durante el anterior procedimiento fue el resultante de haber eliminado todas las malezas menos Lolium. Esta decisión fue tomada debido a que dicha maleza presenta los peores resultados, además al utilizar un conjunto de datos que solamente contuviera Lolium permitió reducir los tiempos de entrenamiento por iteración de 2 minutos a menos de 15 segundos, haciendo viable el cómputo de las 130 iteraciones.

En la figura 6 se grafica el valor función de costo, según el número de iteraciones. Mientras que en la figura 7 el valor de la función de costo varía según el valor adoptado por cada hiperparámetro en sucesivas iteraciones.

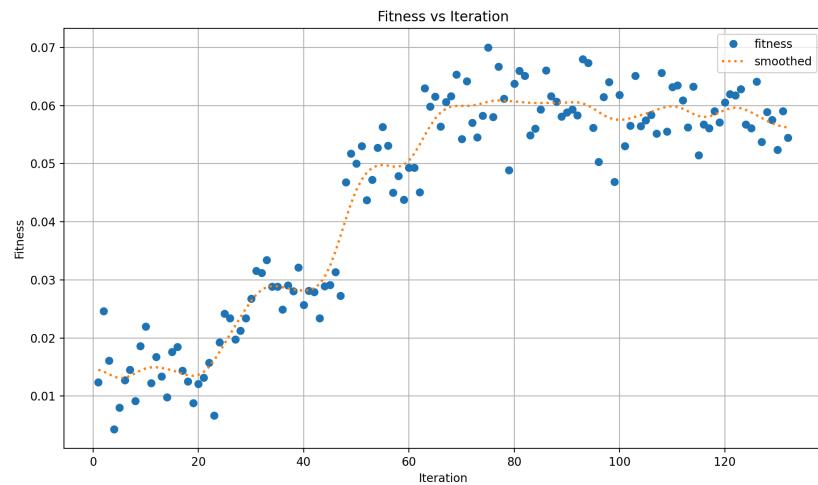


Figura 6- Evolución de la función de costo en las iteraciones de la optimización de hiperparametros

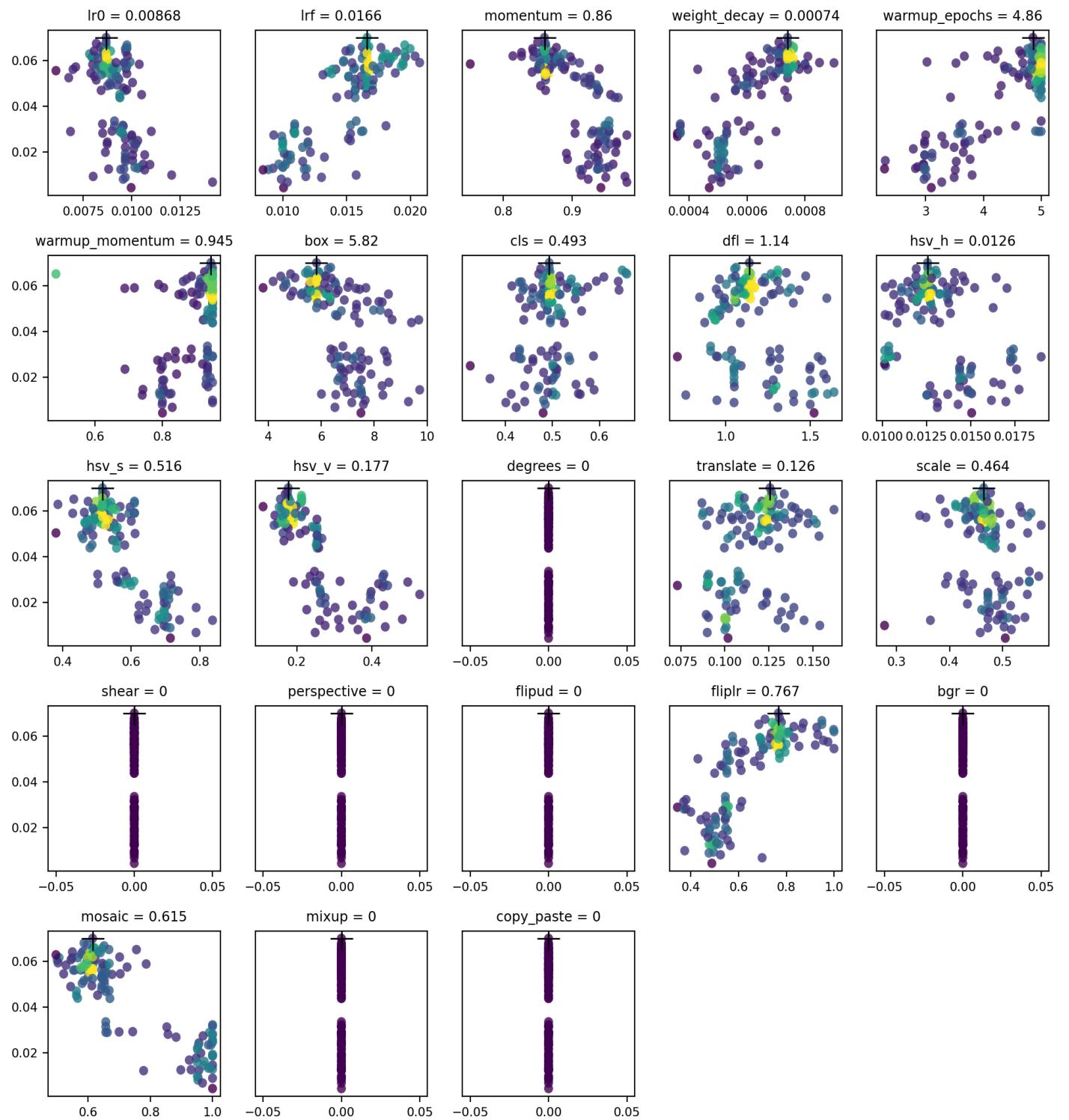


Figura 7 - Evolución de la función de costo según cada hiperparámetro durante la optimización de hiperparámetros

Es importante observar que el número de épocas es únicamente 30 por iteración, el mayor problema que tiene esta decisión es que como el modelo continúa mejorando su desempeño luego de la 30^a época, no necesariamente se encontrará los hiperparámetros que optimiza el rendimiento del modelo, sino aquel que lo hace en las primeras 30 épocas.

Se conjeturó que el rendimiento del modelo podía ser mayormente representado en sus primeras 30 épocas, de forma tal que los modelos con mayor rendimiento también deberían serlo en lo que respecta a sus primeras 30 épocas. En caso de disponer de mayor nivel de cómputo, es recomendable aumentar el número de épocas, lo que hará la selección de hiperparámetros más precisa.

Como es esperable, la anterior suposición no se cumple en todos los casos, algunos ejemplos serán mencionados en la sección [Entrenamiento del modelo definitivo](#), pero de todas formas sí se logró obtener una configuración de hiperparámetros que acelera los tiempos de entrenamiento, y finalmente conducir al entrenamiento del modelo que obtuvo el mayor rendimiento.

Entrenamiento del modelo definitivo

Hasta el momento se han mencionado diferentes técnicas que han sido probadas, en particular:

- Aumento del conjunto de datos utilizando redes generativas
- Optimización de hiperparametros para Lolium
- Uso del congelamiento en las primeras capas (primeras 10) que conforman lo que se conoce como Esqueleto de la red.

Aunque todas las anteriores técnicas fueron introducidas con el fin de mejorar el desempeño de la red, no existen garantías que lo logren, es por ello que se han realizado numerosos entrenamientos, variando las técnicas aplicadas y verificando si las mismas son beneficiosas.

En todos los casos los modelos han sido entrenados como máximo por 200 épocas, en aquellos casos donde se detectaba sobreajuste, el entrenamiento era detenido. Se entiende que un modelo se encontraba en sobreajuste cuando el valor su mAP con confianza 50% o la mAP con confianza 50% hasta 95% disminuye sostenidamente.

A continuación se reportan las ejecuciones realizadas, para cada una se presentará el valor máximo de mAP de confianza 50% y rango 50% a 95% obtenidos según conjunto de validación, también serán añadidos los porcentajes con los que el modelo correctamente predijo cada una de las malezas según el conjunto de testeo. Este último valor es extraído de los componentes de la diagonal de la matriz de confusión.

¿Se congelan las primeras diez capas?	¿Aumenta instancias con redes generativas?	¿Utiliza hiperparametros optimizados?	mAP50	mAP50-95	Precisión Lolioum	Precisión Cyperus	Precisión Ipomoea
NO	NO	NO	0,61	0,43	44%	50%	85%
NO	NO	SI	0,58	0,36	38%	55%	90%
NO	SI	NO	0,54	0,36	34%	44%	85%
NO	SI	SI	0,58	0,38	32%	50%	84%

Si	SI	SI	0,54	0,37	35%	43%	85%
SI	NO	NO	0,56	0,35	36%	49%	86%

De los anteriores resultados resaltamos:

1. Aumentar el conjunto de datos con la técnica de redes generativas antes descrita es perjudicial
2. Aunque el congelamiento de las diez primeras capas aumenta la velocidad con la que el modelo incrementa su rendimiento (en término de mAP), no se obtiene una mejora al finalizar el entrenamiento al compararlo al entrenamiento sin congelamiento.
3. Continuando el anterior apartado, como se visualiza en Figura 8, los modelos que utilizan congelamiento en sus primeras épocas presentan un aumento en el rendimiento más pronunciado, pero poco después su rendimiento se estanca. Además no es claro si el uso de los hiperparametros encontrados incrementa o no la velocidad de aprendizaje, ya que cuando no se aplicó la técnica de congelamiento, el uso de los hiperparametros fue contraproducente, pero cuando el congelamiento si fue aplicado fue beneficioso.

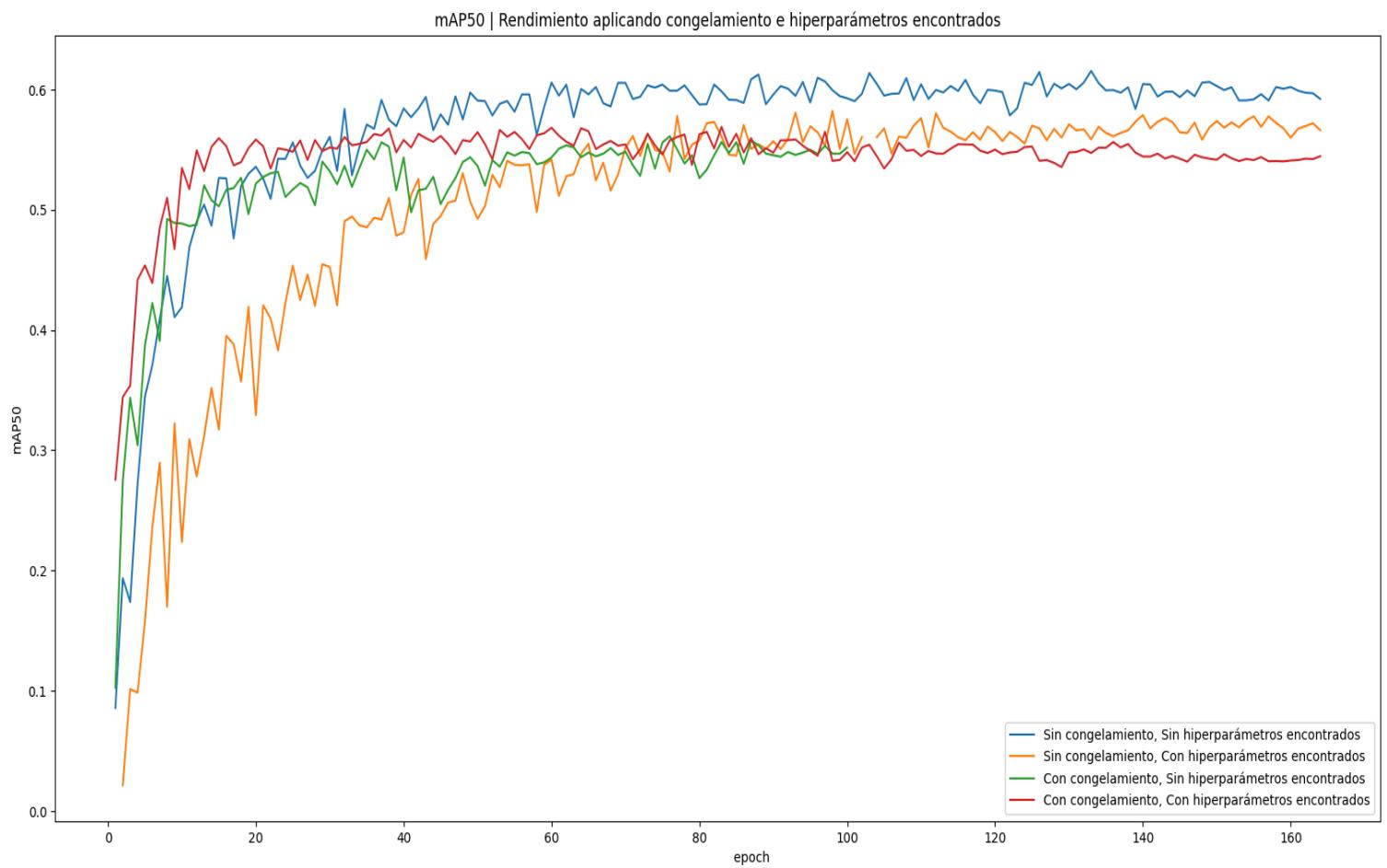


Figura 8 - mAP en confianza 50% en los modelos con congelamiento y uso de hiperparametros optimizados.

4. Aunque sea un contrasentido, el uso de hiperparametros optimizados para maximizar el rendimiento del Lolium, en realidad provocan que el resto de las malezas obtenga mejores resultados, menos Lolium.

Finalmente, ¿cuál de los modelos entrenados debería ser elegido? Observar que si comparamos únicamente los resultados obtenidos utilizando el conjunto de testeo, los dos primeros modelos reportados son los que han obtenido mejores resultados. Donde si se tiene preferencia por la detección de Lolium se debe escoger el modelo que no utiliza los hiperparametros encontrados, ni redes generativas, ni congelamiento; en caso contrario que la preferencia sea por una de las otras dos malezas, se debería escoger el modelo que si hace uso los hiperparametros encontrados, pero no de las redes generativas ni del congelamiento.

En el caso que no haya preferencia por ninguna maleza, se debería escoger el modelo que obtenga la mayor mAP, en el nivel de confianza más alto posible, lo anterior es equivalente a utilizar el indicador F1 en vez de la mAP. Para ambas redes 0,58 es el máximo valor de F1 variando la confianza. Aunque es el modelo entrenado utilizando los hiperparametros escogidos el que obtiene dicho valor en niveles de confianza de 37,7%; mientras que el otro modelo candidato lo obtiene en confianza 26,4%. Al poder obtener iguales resultados pero con un mayor nivel de confianza, **se escoge como modelo final al segundo candidato, o sea, al que utiliza los hiperparametros encontrados, no utiliza congelamiento ni el aumento del volumen del dato por redes generativas.**

EVALUACIÓN DEL MODELO

Modelo final

En esta sección se presentará el modelo final, veremos las características que posee junto con un análisis de las métricas obtenidas, se verán los tiempos de inferencia del modelo y los niveles de confianza para cada una de las plantas.

Como se vió en la sección anterior el modelo final corresponde al modelo con hiperparametros encontrados, sin utilizar congelamiento y sin el aumento de las GAN.

Métricas

La primera métrica a analizar es la confianza F1 para cada planta. En la Figura 9, se puede observar la diferencia que hay para niveles de confianza bajos. Ipomoea, para niveles de confianza bajos, obtiene una F1 por encima de 0.6, mientras que Cyperus y Lolium obtienen valores muy por debajo. Esto tiene sentido, ya que, con un nivel de confianza bajo para etiquetar, se está etiquetando background (pastos o hierbas que no forman parte de las malezas) que no deberían ser etiquetados. Dado que Ipomoea tiene una forma distintiva y difiere en gran medida del background, alcanza niveles altos de F1 con poca confianza. En contraste, Cyperus y Lolium, al ser más susceptibles a confusiones con el background por su forma, alcanzan niveles bajos de F1 con baja confianza.

Entonces, los niveles bajos de confianza nos ofrecen una forma de interpretar qué tanto se confunde cada planta con el background. En contraposición, niveles altos de confianza harán que algunas plantas en las imágenes no se clasifiquen porque están por debajo del nivel de confianza estipulado.

El mejor valor obtenido de F1 es 0.58, alcanzado con niveles de confianza del 37.7%. Un valor máximo de F1 de 0.58 sugiere que el modelo aún tiene margen para mejorar. Aunque 0.58 es el mejor valor obtenido, no es un valor particularmente alto.

Si se quiere sacar el mejor provecho del modelo se puede colocar el parámetro de confianza de detección en 37.7%, esto tendrá un aumento en los falsos positivos, que dependerá del problema en el que se aplique, en el contexto de este proyecto parece tener sentido ya que falsos positivos no tienen un impacto negativo significativo.

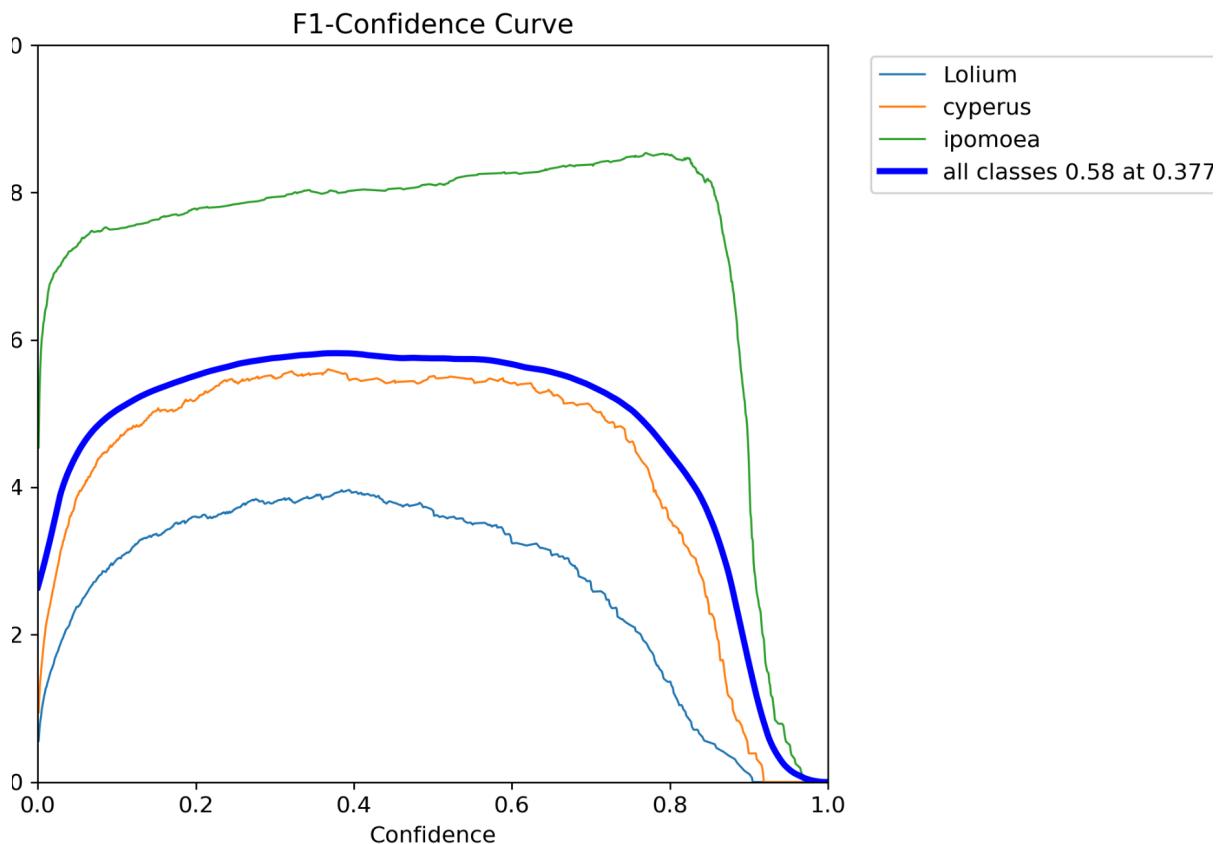


Figura 9, curvas de confianza de métrica F1 por clase y general, indican la precisión del modelo (tanto en verdaderos positivos como en verdaderos negativos) a la hora de evaluar con distintos niveles de confianza.

La Figura 10 corresponde a la matriz de confusión del modelo, allí se puede apreciar aún mejor los grandes niveles de confusión de cyperus y Lolium con Background. Lolium alcanza niveles de hasta el doble de confusión con Background, llegando hasta el 62%. Esto es, cuando verdaderamente es Lolium etiqueta como Background un 62% de las

veces, mientras que en contraposición cuando verdaderamente era Background lo etiqueta como Lolium un total de 46%.

En el contexto del problema esto representa aplicar pesticida en lugares donde no lo requería, es decir falsos positivos.

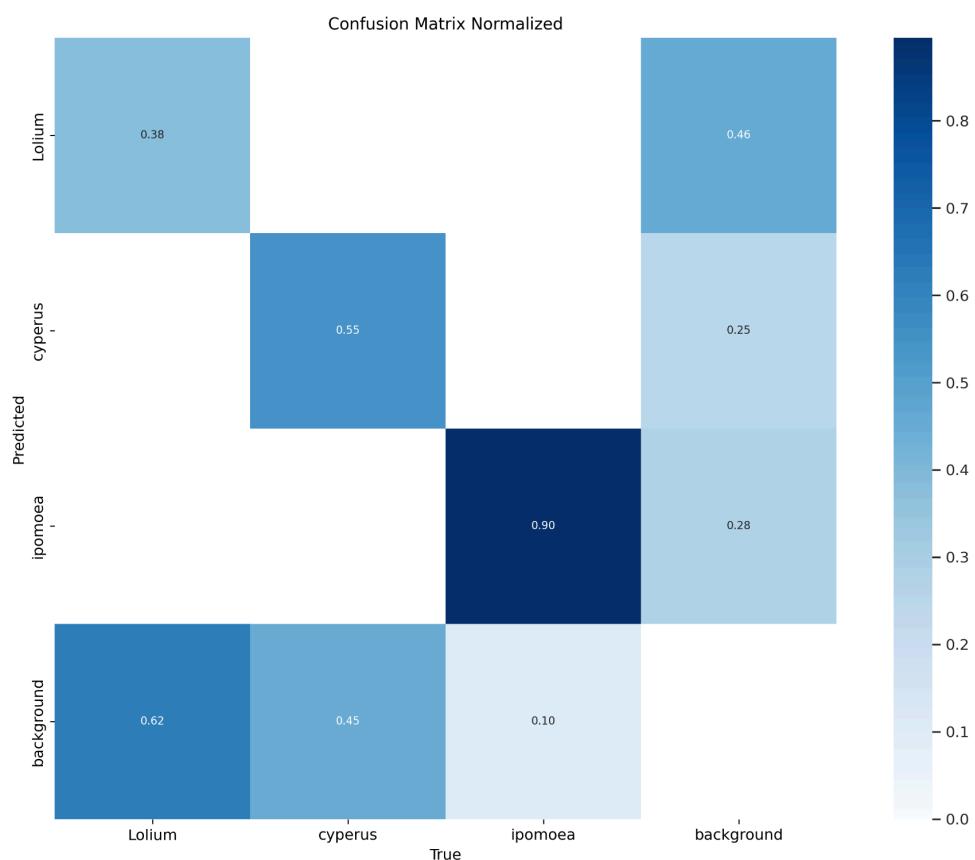


Figura 10

Tiempo de inferencia

El modelo YOLOv9c es el penúltimo de mayor tamaño en cuanto a necesidades computacionales de los que ofrece la implementación de ultralytics. Por lo tanto, se evaluará el tiempo que requiere inferir una imagen a dicho modelo. Como se puede observar en la figura 11, el modelo YOLOv9c posee 25.5 millones de parámetros y requiere una capacidad computacional de 102.8 FLOPs (Operaciones de Punto Flotante por Segundo) del inglés,

Floating Point Operations per Second. Esta medida nos informa sobre la capacidad de cálculo requerida de la unidad de procesamiento para que el modelo funcione correctamente.

Modelo	tamaño (píxeles)	mAPval 50-95	mAPval 50	parámetros (M)	FLOPs (B)
YOLOv9t	640	38.3	53.1	2.0	7.7
YOLOv9s	640	46.8	63.4	7.2	26.7
YOLOv9m	640	51.4	68.1	20.1	76.8
YOLOv9c	640	53.0	70.2	25.5	102.8
YOLOv9e	640	55.6	72.8	58.1	192.5

Figura 11, tabla de requerimientos de las implementaciones de ultralytics según modelo de yolo

El modelo mencionado se evalúa sobre todo el conjunto de test final (63 imágenes) y se toma el tiempo promedio de inferencia, el cual resulta en 0.0554 segundos en una gráfica T4 de google collaboratory y de 0.068 milisegundos en la gráfica AMD 6700 xt, consideramos esto un buen tiempo de inferencia ya que se tienen en promedio 18.05 y 14.70 inferencias por segundo, lo cual consideramos suficiente para el contexto del problema.

CONCLUSIONES

A partir de los experimentos y resultados obtenidos, se concluye que fue posible hacer un modelo que, si bien comparando con otros trabajos a nivel de mAP es inferior, consideramos que los resultados son aceptables, teniendo en cuenta que además de que el equipo no contaba con previo conocimiento del área, no se contaba con un conjunto de datos acorde al problema y este fue armado sin definir un criterio basado en expertos o metodologías sobre el área de estudio. Se obtienen las siguientes conclusiones del entrenamiento:

Impacto del Aumento de Datos con Redes Generativas:

- El aumento del conjunto de datos utilizando redes generativas con los métodos antes descritos no ha sido beneficioso para mejorar el rendimiento del modelo. De hecho, en varios casos, esta técnica ha resultado perjudicial, lo que sugiere que las instancias generadas no eran suficientemente representativas o introducían ruido al modelo.

Congelamiento de Capas Iniciales:

- El congelamiento de las primeras diez capas de la red (esqueleto de la red) mostró un aumento en la velocidad de aprendizaje inicial del modelo. Sin embargo, esto no resultó en un rendimiento superior al final del entrenamiento. Los modelos con capas congeladas tendían a estancarse en su rendimiento tras las primeras épocas.

- Esta técnica presentó una mejora considerable del modelo respecto al tiempo que insume al modelo llegar a resultados aceptables.

Optimización de Hiperparámetros:

- El uso de hiperparámetros optimizados para *Lolium* tuvo un impacto mixto. Al contrastar el uso de los mismos con el modelo sin congelamiento de capas, ni redes generativas (es decir, el mejor modelo obtenido), se observa que los resultados son exactamente el contrario de lo esperado, presentando mejoras a la hora de medir la precisión en las especies *Cyperus* e *Ipomoea*, pero afectando a la especie *Lolium*, que recordemos fue con la que se entrenaron dichos hiperparámetros.

Finalmente, compartimos la figura 12 que es una imagen clasificada por el modelo perteneciente al conjunto de validación, donde la imagen original etiquetada de forma manual no había incluido el *Lolium* de la izquierda por un simple error humano, lo interesante es que el modelo sí logró reconocer el blur por sí solo.



Figura 12

TRABAJO FUTURO

Presentamos a continuación las posibles líneas a seguir investigando que consideramos que podrían aportar resultados interesantes.

- **Generación de instancias mediante redes generativas**

- El mayor problema a la hora de utilizar el modelo CycleGAN para la generación de nuevas instancias, es encontrar un filtro apropiado para conseguir instancias pertinentes al contexto para la mejora del modelo. En nuestro caso, utilizamos un filtro de redes generativas entrenado con imágenes de estaciones cuyas instancias incluían características que no se obtendrían normalmente en la ubicación de nuestro problema (por ejemplo, nieve). Se podría, si se halla un nuevo modelo publicado o se entrena un nuevo modelo, tratar de utilizarlo para mejorar la generalización del modelo tanto para las estaciones del año como para otro tipo de conceptos.
- Otro paradigma dentro del aumento de datos dentro de las redes generativas es la transferencia de conocimiento mediante el etiquetado de instancias utilizando diferentes clasificadores de detección de objetos, se podría llevar a cabo entonces con un modelo generativo (posiblemente similar a StyleGAN) que se encargue de generar nuevas instancias y estas se etiqueten utilizando algún algoritmo del tipo mencionado o bien se etiqueten manualmente.

- **Posibles mejoras al conjunto de datos**

- Mejorar la metodología de obtención de imágenes
 - La toma de imágenes podría realizarse por parte del equipo investigador para relevar la mayor cantidad posible de variaciones de las especies solicitadas, tanto de las mismas en diferentes estaciones del año como de diferentes etapas de la vida de la planta. Nótese que en el caso de un detector de malezas con láseres, mayormente interesaría implementar un dataset que detecte los brotes que deberían ser erradicados.
 - Como ya se mencionó previamente, se podría llevar a cabo el etiquetado de las instancias del dataset con más cuidado, ayudado de expertos que definan cual es la parte de interés a detectar. Se podrían definir criterios profesionales basados en conocimiento sobre las especies, esto permitiría detectar mejor las partes más características de dichas especies o realizar un etiquetado con menos variaciones de criterios que ayude a mejorar las métricas de precisión.

- **Posibles mejoras al proceso de entrenamiento**

- Para poder obtener mejores resultados, se recomienda realizar un procedimiento de mejora de hiperparámetros sobre los modelos a utilizar, en este trabajo se priorizó el minimizar el tiempo de uso de los recursos para obtener resultados de forma ágil, se podría mejorar este procedimiento utilizando el dataset completo a la hora de ejecutar dicho proceso decrementando el tiempo de ejecución de cada modelo mediante el congelamiento de las diez primeras capas del modelo, el cual como ya vimos

consigue resultados aceptables en una cantidad mucho menor de épocas que sin congelamiento de dichas capas, lo cual implica una mejora significativa de tiempo de ejecución al explorar gran cantidad de modelos.

GLOSARIO

1. **Hiperparámetro**: Son parámetros cuyos valores son utilizados para controlar un proceso de aprendizaje automático. Los otros parámetros del proceso son los que se buscan optimizar mediante el entrenamiento.
2. **mAP** (En inglés Mean Average Precision): Dentro del problema de Detección de Objetos se ha diseñado la métrica mAP, intentando cuantificar la correctitud con la que un modelo predice la ubicación y clase de los objetos buscados dentro de una imagen. Dado un nivel de confianza especificado (o un rango), el valor de la mAP será el promedio de la AP de cada clase. La notación mAP50 representa el valor de mAP dada una confianza de 50%, mientras que mAP50-95 corresponde al rango de confianza entre 50% y 95%.
3. **AP** (Precisión media, en inglés Average Precision): Dada una clase y una confianza, la AP será el área debajo de la curva PR (precisión y recuperación) para las predicciones realizadas por el modelo sobre dicha clase. El cálculo de la curva PR es realizado sobre los falsos positivos y los verdaderos positivos. Las predicciones son catalogadas como falsas o verdaderas según el valor de su IoU y la confianza establecida, de forma que las predicciones verdaderas serán aquellas cuyo IoU sea mayor o igual a la confianza.
4. **IoU** (Intersección sobre unión, en inglés intersección over unión): dadas dos áreas (generalmente una predicha y otra real), el su IoU será la cociente entre el área de su intersección sobre la de su unión.

BIBLIOGRAFÍA

Dang, Fengying, et al. "YOLOWeeds: A novel benchmark of YOLO object detectors for multi-class weed detection in cotton production systems." *sciencedirect*, 6 January 2023, <https://www.sciencedirect.com/science/article/pii/S0168169923000431>.

Accessed 9 June 2024.

GBIF. "GBIF.org." *GBIF.org*, <https://www.gbif.org/>. Accessed 24 June 2024.

"Global Wheat." *Global WHEAT Dataset*, <https://www.global-wheat.com/>. Accessed 24 June 2024.

Hammami, M., et al. "CYCLE GAN-BASED DATA AUGMENTATION FOR MULTI-ORGAN DETECTION IN CT IMAGES VIA YOLO. International Conference on Image Processing (ICIP)." *IEEEXPLORE.ieee.org*, October 2020, <https://ieeexplore.ieee.org/document/9191127>. Accessed 9 June 2024.

HUANG, QIHAN, et al. "A Survey of Deep Learning for Low-Shot Object Detection." *A Survey of Deep Learning for Low-Shot Object Detection*, 25 Octubre 2023, <https://arxiv.org/pdf/2112.02814.pdf>. Accessed 11 6 2024.

Wang P., Wang, and Tang Tang Y.. *Wang P, Tang Y, Luo F, Wang L, Li C, Niu Q and Li H (2022) Weed25: A deep learning dataset for weed identification. Front. Plant Sci. 13:1053329. doi: 10.3389/fpls.2022.1053329.*