Segmentación de Áreas Afectadas en Hojas de Plantas para la Detección de Enfermedades

Nicolás Redondo
nicoredondoo@correo.ugr.es

Carlos Bermúdez
carlosbermudez@correo.ugr.es

Guillermo Ramblado
guilleramblado@correo.ugr.es

Pablo Ramblado
pabloramblado@correo.ugr.es

Abstract

Este proyecto busca desarrollar un modelo que detecte y segmente automáticamente las áreas afectadas en hojas de plantas mediante imágenes de alta resolución y sus máscaras correspondientes. El modelo permite identificar visualmente las zonas dañadas y calcular el porcentaje de área infectada en cada hoja, proporcionando una herramienta de diagnóstico visual que optimiza el manejo agrícola.

El trabajo se centra en la implementación de la red U-Net, ampliamente reconocida por su eficacia en tareas de segmentación, con el objetivo de automatizar la identificación de áreas afectadas y facilitar una evaluación precisa del impacto en las hojas. Para ello, se llevaron a cabo experimentos empleando diversas configuraciones de U-Net con backbones como ResNet34 y XResNet34. El rendimiento del modelo se evaluó mediante métricas clave como Accuracy, Dice y Recall.

El conjunto de datos utilizado, Leaf Disease Segmentation Dataset, incluye imágenes de hojas con sus respectivas máscaras de zonas infectadas, lo que permitió entrenar y validar modelos robustos para la segmentación. Los resultados obtenidos muestran una alta precisión en la detección de áreas afectadas, superando significativamente los métodos manuales en eficiencia y consistencia.

Este trabajo no solo valida la efectividad del modelo U-Net en el diagnóstico agrícola, sino que también introduce soluciones basadas en inteligencia artificial como herramientas clave para una agricultura más eficiente y sostenible, promoviendo un manejo optimizado de los cultivos y una mejor toma de decisiones en el sector agrícola.

1. Introducción

Este proyecto se centra en la segmentación automática de áreas afectadas en hojas de plantas. El objetivo es desarrollar un modelo de aprendizaje profundo que sea capaz de detectar y segmentar con precisión las zonas infectadas en imágenes de alta resolución de hojas, junto con sus máscaras correspondientes. Este enfoque tiene una gran relevancia en el contexto agrícola, ya que permite una identificación más eficiente y precisa de las áreas dañadas, lo que puede facilitar el diagnóstico y tratamiento de las plantas.

La motivación detrás de este trabajo es la creciente necesidad de herramientas automatizadas para la detección y monitoreo de enfermedades en las plantas. Actualmente, la identificación de áreas afectadas se realiza de manera manual, lo cual es un proceso laborioso, propenso a errores y que consume mucho tiempo. La automatización mediante técnicas de visión por computador no solo mejora la precisión, sino que también puede contribuir a optimizar los recursos y reducir el uso de productos químicos, haciendo la agricultura más sostenible.

El modelo que se utiliza en este proyecto es **U-Net**, una red neuronal ampliamente reconocida por su eficacia en tareas de segmentación. Se busca adaptar este modelo para segmentar con precisión las áreas afectadas de las hojas, asegurando que el sistema funcione eficientemente y pueda ser utilizado en escenarios agrícolas reales, ofre-

ciendo así una herramienta de diagnóstico visual confiable para el manejo de enfermedades de las plantas.

2. Contexto

Aparte de todo lo visto en clase de teoría, lo único que necesitamos conocer es la red neuronal **U-Net**. Esta es una convolucional ampliamente utilizada para la segmentación de imágenes. Originalmente fue desarrollada para segmentar imágenes biomédicas, su arquitectura en forma de ”U” permite procesar imágenes con alta precisión. Además de su aplicación en el ámbito médico, esta se emplea en cartografía satelital y en **la agricultura de precisión**, donde facilita a los robots identificar y segmentar áreas específicas de cultivos para optimizar tareas agrícolas. **Es por ello que hemos decidido emplearla en la segmentación de enfermedades en las hojas.** Como dato curioso, U-Net es utilizada como base en DALL-E, la famosa aplicación de OpenAI, para la generación de imágenes.

Un autocodificador clásico sigue una arquitectura donde la información de entrada se reduce progresivamente a través de capas de compresión y luego, en la fase de decodificación, esta se expande gradualmente hasta reconstruir la salida con el mismo tamaño que la entrada original. Sin embargo, esto sigue un enfoque lineal el cual puede limitar la capacidad del modelo para capturar y transmitir todas las características relevantes de los datos, lo que puede llevar a una pérdida de información.

En cambio, U-Net con su forma de ”U” resuelve este problema. Durante la fase de decodificación, utiliza operaciones de deconvolución o transposición de convoluciones para recuperar el tamaño original de la imagen, pero con una diferencia importante: combina las características comprimidas con información detallada de las etapas iniciales de la red. Esto permite conservar más características importantes, evitando así el problema del cuello de botella típico de los autocodificadores y mejorando la precisión de la segmentación.

2.1. Arquitectura U-Net

Como hemos dicho y se puede observar, esta se trata de un conjunto de capas de convolución y de “max pooling” que permiten crear un mapa de características de una imagen, reduciendo el tamaño para disminuir el número de parámetros de la red.

Su arquitectura es la siguiente:

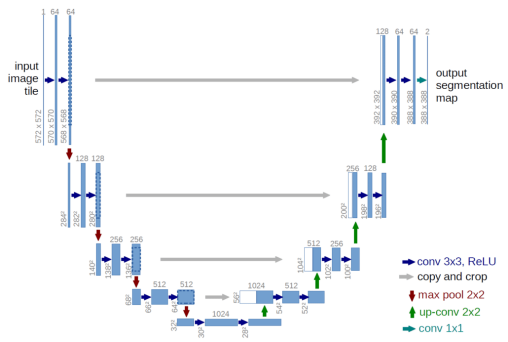

Figure 1. Arquitectura de la red U-Net

La arquitectura de U-Net se compone de dos partes principales: una descendente y otra ascendente. La **parte descendente (encoder)** está formada por capas de convolución combinadas con reducciones de resolución (*downsampling o pooling*), encargándose de extraer la información semántica de la imagen y sus diferentes componentes, como las áreas relevantes de la hoja. La **parte ascendente (decoder)**, por su parte, reconstruye el mapa de segmentación semántica utilizando capas de convolución y aumentos de resolución (*upsampling*), para reconstruir la imagen, añadiendo detalles y estructuras, como la identificación precisa de las zonas afectadas por la enfermedad.

Las **conexiones de salto** en U-Net son fundamentales para mejorar la precisión en la segmentación de imágenes. Estas conexiones enlazan directamente capas correspondientes en el codificador y decodificador, permitiendo la transferencia de información detallada desde las primeras etapas de la red hacia las últimas. De esta forma se ayuda a preservar características espaciales importantes que podrían perderse durante la reducción de resolución en el codificador. Además, las conexiones de salto facilitan un mejor flujo de gradiente durante el entrenamiento, mejorando la eficacia de la retropropagación y evitando el problema del gradiente desaparecido.

Una característica distintiva de U-Net es el uso de un **gran número de canales de características durante la**

fase ascendente, lo que facilita la propagación de la información contextual (*relaciones espaciales y estructurales de los elementos dentro de una imagen que ayudan a comprender su significado global*) a las capas de mayor resolución. Esto permite que los detalles importantes extraídos durante la fase descendente se mantengan y se refinen al reconstruir la imagen.

Gracias a esta estructura, la fase ascendente es muy similar a la fase descendente, creando una simetría con forma de "U" que facilita una reconstrucción precisa de la imagen. Además, U-Net evita el uso de capas completamente conectadas, lo que mejora la eficiencia computacional.

3. Métodos

Hay que tener en cuenta que para cada imagen, se tiene su respectiva máscara, la cual indicará que píxeles corresponden a enfermedad.

Figure 2. Imagen 1 y Máscara 1

Aunque a simple vista las máscaras parecen tener solo colores negro y blanco, nuestro código revela que la mayoría contienen valores únicos como 0 y 38, que corresponden al negro y blanco, respectivamente. Sin embargo, muchas otras máscaras incluyen valores adicionales fuera de estos dos en el rango [0-38], lo que sugiere la presencia de tonos grises que, **aunque apenas son perceptibles visualmente, pueden influir en los resultados al hacer uso de ellas**. Esto indica que las máscaras no se limitan a representar sólo las áreas de enfermedad y no enfermedad, sino que también contienen píxeles con valores intermedios. Por lo tanto, **es necesario convertir estas máscaras a un formato binario**, donde todo píxel coloreado se considere blanco con el valor "1" (*infectado*) y el resto negro (*no infectado*) con el valor "0", lo que simplifica y mejora la segmentación.

Podemos notar que, en los resultados obtenidos, **los valores atípicos se muestran en rojo**, aunque estos cambios no son muy perceptibles visualmente y las diferencias son mínimas, la presencia de estos pequeños cambios puede influir en el proceso de segmentación, afectando la precisión y proporcionando resultados menos efectivos.

Figure 3. Atípicos de la Máscara 1 marcados

Todo este proceso de forma similar, lo realizaremos con las máscaras de las hojas segmentadas.

Como ya se ha hecho en anteriores prácticas, hemos creado dentro del directorio /data (donde se almacenan las imágenes originales sin transformaciones) dos nuevos directorios,una carpeta para entrenamiento /train y otra para prueba /test. Esta división se ha hecho teniendo en cuenta que un **10% será para prueba y el 90% para entrenamiento.**

Además, se realizaron varias pruebas para mejorar la segmentación de las hojas y obtener un porcentaje más preciso de la enfermedad, considerando que nuestro dataset presenta hojas en entornos complejos, rodeadas de otras hojas o con fondos variados. Esto hace que la segmentación sea más complicada, pero también más realista para aplicaciones agrícolas reales.

En las primeras pruebas, se intentó separar la hoja del fondo utilizando los **canales RGB y HSV**, pero no encontramos un canal consistente que funcionara para todas las imágenes, debido a la variabilidad del fondo. Posteriormente, se probó detectar el contorno más grande de la hoja y aplicar técnicas morfológicas para segmentarla, lo que mejoró los resultados en algunas imágenes.

Finalmente, aunque se podría haber optado por un proyecto más complejo utilizando un modelo automático de segmentación, las diferencias significativas entre las imágenes y la complejidad añadida de estos modelos nos

llevaron a centrarnos en el objetivo principal: **segmentar la hoja en sí misma**. Debido a estas complicaciones, se decidió realizar un **recorte manual** de las imágenes, y, posteriormente, entrenar un modelo (Usando el *Modelo 1* que veremos posteriormente) para calcular el porcentaje de área afectada.

3.1. Modelos

A lo largo de este proyecto se han probado distintos modelos utilizando el **DataBlock** creado con MaskBlock para segmentación binaria. Este bloque de datos gestiona las imágenes y sus máscaras asociadas, realizando transformaciones como el recorte aleatorio (*RandomResizedCrop*) y transformaciones de lote (*aug_trasnforms*) asegurando que las imágenes sean de tamaño 256x256.

Como *backbone* (codificadores para el modelo), hemos optado por emplear una de las arquitecturas que nos proporcionaba internamente FastAi : **Resnet34**. Esta red se basa principalmente en el uso de *conexiones residuales*, que resuelven el problema asociado a la degradación del rendimiento del modelo al aumentar su profundidad.

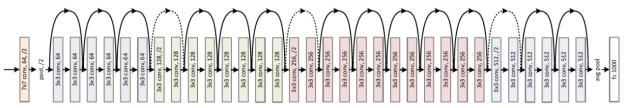

Figure 4. Arquitectura de Resnet34

Su nombre proviene justamente de las 34 capas convolucionales que emplea dicha red, agrupadas en bloques residuales, conectando la salida de cada bloque con la entrada del siguiente, tal y como se muestra en la imagen 4 .

4. Experimentos

Como **función de pérdida** hemos usado: **BCEWithLogitsLoss**. Esta combina **Binary Cross-Entropy (BCE)** con una capa de **sigmoide** de manera eficiente.
En problemas de segmentación binaria, como el nuestro, las salidas del modelo son *logits* (valores continuos), que no están directamente entre 0 y 1. La función *BCEWithLogitsLoss* aplica internamente la función sigmoide a estos *logits* para convertirlos en probabilidades, realizando posteriormente una umbralización, facilitando así la comparación con las etiquetas binarias (0 o 1) que representan la clase de fondo o la clase de interés (enfermedad en la hoja), y calculando la pérdida mediante la *entropía cruzada*.

Además, para hacer frente al desbalanceo de clases presente a la hora de segmentar las zonas infectadas de la hoja, se ha optado por emplear el parámetro **pos_weight** que nos proporciona la función de pérdida previa, y que nos permite asignar un mayor peso a la clase minoritaria a la hora de calcular la pérdida, en este caso, la clase *infectada*, penalizando más los falsos negativos. Esto es, no es tan relevante el hecho de clasificar un píxel no infectado como infectado (falso negativo), aunque sí que es importante que el modelo sepa detectar con la mayor precisión posible la existencia de zonas infectadas.

Las dos **métricas** utilizadas han sido las siguientes, cabe detectar que todas estas se han implementado manualmente, ya que las que proporcionaba *fastai* no se adaptan bien a nuestra salida del modelo:

- **Accuracy:** Mide la proporción de píxeles correctamente clasificados, tanto enfermos como no enfermos. Esto nos proporciona una visión general de cómo el modelo está funcionando en términos de clasificación de toda la imagen, sin centrarse específicamente en las áreas de interés. Sin embargo, en nuestro problema con clases desbalanceadas, esta métrica puede ser engañosa, ya que puede estar influída por la correcta clasificación de píxeles no enfermos, que son más abundantes que las áreas afectadas.

- **Dice:** Mide la superposición entre las predicciones y las etiquetas reales de las áreas relevantes, como las zonas afectadas de la hoja. Esta métrica es especialmente útil en segmentación, ya que se enfoca en las áreas de interés, ignorando el fondo y concentrándose en las zonas que el modelo identifica como enfermas. Un valor alto indica una segmentación precisa de las áreas afectadas.

- **Recall:** Mide la capacidad del modelo para identificar correctamente todas las zonas relevantes (las áreas enfermas de la hoja), al comparar las predicciones con

las etiquetas reales. En otras palabras, el recall evalúa qué tan bien el modelo captura todas las zonas que deberían haberse clasificado como enfermas, sin preocuparse por las áreas no afectadas. Un valor alto de recall indica que el modelo está detectando la mayoría de las zonas enfermas.

Se ha usado un **tamaño de batch** de **16**. Usar un valor pequeño implica que el modelo se actualizará con mayor frecuencia durante el entrenamiento, ayudando a una mejor generalización al capturar más variaciones de los datos. Sin embargo, esto también significa que el proceso de entrenamiento puede tardar más tiempo en completarse, ya que se realizan más iteraciones por época.

4.1. Dataset

Se ha utilizado el dataset **Leaf Disease Segmentation Dataset** para entrenar y evaluar los diferentes modelos. Este dataset es público dentro de la web Kaggle y está específicamente diseñado para la segmentación de enfermedades en hojas. Proporciona un conjunto detallado de imágenes de hojas junto con sus correspondientes máscaras, lo que lo hace altamente útil para tareas de segmentación

El conjunto de datos contiene un total de 588 imágenes, cada una de ellas acompañada por su máscara, que identifica las áreas afectadas por enfermedades. Las imágenes del dataset incluyen una variedad de condiciones, capturando hojas en diferentes estados y con distintos tipos de enfermedades, lo que lo convierte en un recurso valioso para el desarrollo y evaluación de modelos de segmentación de imágenes.

Las máscaras proporcionadas facilitan la tarea de segmentación, permitiendo que los algoritmos aprendan a distinguir con precisión las áreas sanas de las afectadas en las hojas. La diversidad de las imágenes y las condiciones representadas en este dataset aseguran que los modelos entrenados sean robustos y puedan generalizar bien a nuevos datos.

Además, hemos creado manualmente las máscaras específicas de las áreas afectadas de cada hoja. Esto permite que el cálculo del porcentaje de infección se realice sobre estas áreas específicas, en lugar de considerar toda la imagen completa. La razón detrás de esta decisión ya ha sido explicada anteriormente, ya que al trabajar con un dataset tan complejo, es muy complicado obtener una máscara precisa de la hoja utilizando distintos métodos automáticos. No se optó por usar un modelo adicional para segmentar la hoja en sí, con el fin de evitar complicar la práctica y alejarnos del objetivo principal de la misma, que es centrarse en la segmentación de las áreas afectadas por la enfermedad.

5. Conclusiones

Los resultados obtenidos del proyecto están representados en términos de métricas como *Loss, Accuracy, Dice y Recall* en la siguiente tabla:

Modelo	Loss	Accuracy	Dice	Recall
Hojas segmentadas - Resnet34	0.233266	0.903190	/	0.657050
Hojas segmentadas - Resnet34 con pesos	0.232712	0.903047	/	0.692462
Segmentación zona infectada - Resnet34	0.391960	0.822960	0.650103	0.828449
Segmentación zona infectada - Resnet34 con pesos	0.249647	0.901009	0.736064	0.695288

Figure 5. Tabla comparativa de los resultados obtenidos

5.1. Modelos y enfoques evaluados

1. **Segmentación de hojas(ResNet34 sin pesos preentrenados):** Este modelo presentó un *accuracy* alto (90.31%), indicando que el modelo logra identificar correctamente la mayoría de las zonas en las hojas. Sin embargo, su *recall* relativamente bajo (65.71%) sugiere que no todas las zonas relevantes son identificadas, lo que podría deberse a una mayor propensión a falsos negativos.

2. **Segmentación de hojas(ResNet34 con pesos preentrenados):** El uso de pesos preentrenados no parece mejorar significativamente el *accuracy* (90.30%), pero sí contribuye a un aumento del recall (69.25%), lo que indica que este modelo es más efectivo a la hora de detectar zonas de interés relevantes, aunque aún con margen de mejora.

3. **Segmentación de zonas infectadas(ResNet34 con pesos preentrenados):** Este modelo, enfocado específicamente en la segmentación de las zonas infectadas, muestra un *accuracy* más bajo (82.30%) en

comparación con los modelos de segmentación de hojas. Sin embargo, su *recall* (82.84%) y su métrica Dice (0.65) evidencian que el modelo logra capturar de manera más completa las zonas infectadas, con una menor tasa de falsos negativos. Esto es crítico para aplicaciones donde es importante no omitir zonas infectadas.

4. **Segmentación de zonas infectadas (ResNet34 con pesos preentrenados):**Incorporar pesos preentrenados para la segmentación de zonas infectadas mejora de manera notable varias métricas clave: el *accuracy* se eleva al 90.10% y la métrica *Dice* aumenta a 0.736, lo que denota un mejor equilibrio entre precisión y sensibilidad. Sin embargo, el *recall* disminuye ligeramente (69.52%), lo que sugiere que aún existen limitaciones para capturar todas las zonas relevantes.

El análisis de los modelos indica que, si bien la pérdida es un factor relevante, la métrica más importante en este contexto es el **Recall**, ya que mide la capacidad del modelo para identificar correctamente todas las áreas enfermas, minimizando los falsos negativos. Bajo esta premisa, el modelo **Segmentación zona infectada - Resnet34 (sin pesos)** es el mejor, ya que logra el mayor Recall (0.828449), lo que lo hace más efectivo en la detección completa de las zonas afectadas, aunque tenga una pérdida más alta. Por otro lado, el modelo **Segmentación zona infectada - Resnet34 con pesos** muestra una mejor pérdida y un valor de Dice superior, indicando mayor precisión en la segmentación general, pero con un Recall menor (0.695288), lo que implica que podría pasar por alto algunas áreas enfermas. En conclusión, ya que nuestra prioridad es detectar todas las zonas enfermas, el modelo sin pesos es la opción más adecuada, mientras que el modelo con pesos ofrece un equilibrio entre precisión y detección.

Podemos observar algunos resultados de nuestro modelo en las imágenes resultantes. Mostramos detalles como el área total de la hoja, el área infectada, el porcentaje de **infección predicho** y el porcentaje de **infección "REAL"**. Como se puede ver, el modelo ha predicho un porcentaje de infección del **18.60%**, mientras que el porcentaje real es del **14.74%**. Esto confirma que el modelo ha logrado un resultado decente, con una cercanía aceptable al valor real. Tal como se discutió en las tablas anteriores, aunque en algunos casos el modelo puede desviarse ligeramente, en otros se acerca lo suficiente, demostrando un rendimiento adecuado en la detección de áreas infectadas.

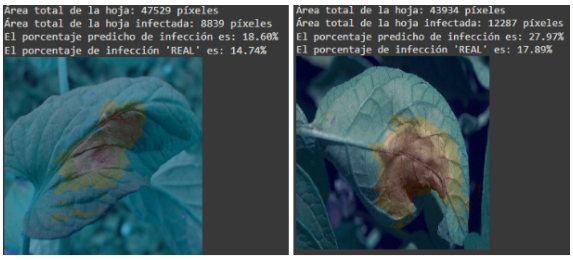

Figure 6. Porcentaje predicho vs Porcentaje Real

5.2. Conclusiones finales

En términos generales, el modelo de segmentación de zonas infectadas con pesos preentrenados es el más equilibrado en esta práctica, logrando un rendimiento destacado en métricas clave como *accuracy* (90.10%) y *Dice* (73,6%). Aunque su *recall* es menor que el del modelo sin pesos preentrenados, la mejora en el resto de métricas sugiere que este enfoque es más adecuado para aplicaciones prácticas donde un equilibrio entre precisión y sensibilidad es crítico.

Es primordial elegir el dataset siendo consecuente con el problema a resolver, ya que nos ha dado una serie de complicaciones que se podrían haber resuelto simplemente con un mayor conocimiento del problema a resolver, ya que como se ha dicho en clase, los datos son una parte fundamental del proceso y una mala elección de estos, o no acordes al problema, complican el procedimiento.

Además, a la hora de segmentar las zonas infectadas de las hojas, optamos por seleccionar el modelo que recibe directamente como entrada las imágenes de las hojas, sin ningún tipo de preprocesamiento previo. Tal y como se aprecia en la tabla de resultados, segmentar primeramente la hoja antes de segmentar la zona infectada con el objetivo de que el modelo se enfoque únicamente en la hoja no nos ha ofrecido una gran mejora con respecto al planteamiento inicial de segmentar directamente la zona infectada. Esto puede ser debido al hecho de que el error que genera el

primer modelo (segmentación de la hoja) es arrastrado hacia el siguiente modelo de segmentación de zonas infectadas, incrementando aún más el error de este último modelo. O quizás, por el hecho de que al tener más imagen, es capaz de aprender mejor determinadas características y llega a generalizar mejor. Ha sido un proyecto que nos ha parecido muy interesante y sobre todo por el hecho de que hemos sido capaces de pensar y desarrollar diferentes formas de poder resolverlo, aplicando no solo los conocimientos y técnicas adquiridas en clase, sino como en la búsqueda de referencias o en las propias pruebas empíricas de nuestro desarrollo.

Para futuros trabajos, sería interesante explorar técnicas de optimización adicionales, como ajustes más finos de hiperparámetros, el uso de arquitecturas más complejas, o el empleo de métodos de aumento de datos para mejorar el recall sin comprometer el resto de métricas.

6. References

1. https://docs.fast.ai/
2. https://github.com/fastai/fastai/blob/main/fastai/vision/learner.py
3. https://www.sciencedirect.com/science/article/abs/pii/S0168169922008195
4. https://www.juansensio.com/blog/050_cv_segmentacion
5. https://rua.ua.es/dspace/bitstream/10045/144220/1/Segmentacion_Binaria_y_Multiclase_para__Al_Shikh_Othman_Nassif_Mohamad_Hazem.pdf
6. https://es.eitca.org/artificial-intelligence/eitc-ai-adl-advanced-deep-learning/advanced-computer-vision/advanced-models-for-computer-vision/examination-review-advanced-models-for-computer-vision/how-does-the-u-net-architecture-leverage-skip-connections-to-enhance-the-precision-and-detail-of-semantic-segmentation-outputs-and-why-are-these-connections-important-for-backpropagation/