Informe final

Categoría: Futuras promesas

Participante: *Aprende Máquina*
Raúl Coterillo (representante), Arkaitz Bidaurrazaga
Email contacto: **rcoterillo@bcamath.org**

BIlbao, 17 de Enero de 2023

Contenido

1 Enfoque

1.1 Reto abordado y principales desafíos

Los campos de la Inteligencia Artificial (IA) y el Machine Learning (ML) han visto avances significativos en los últimos años, particularmente en el área de reconocimiento y clasificación de imágenes. Concretamente, el reto planteado en esta competición trata sobre la detección y cuantificación de enfermedades en cultivos utilizando imágenes aéreas de drones equipados con cámaras multiespectrales.

Los cultivos dañados por enfermedades pueden tener un impacto devastador en la agricultura, traduciéndose en pérdidas significativas en cuanto a cosechas e ingresos. Una pronta detección de dichas enfermedades es crucial para mitigar su efecto e implementar medidas de control. Sin embargo, detectarlos manualmente requiere mucho tiempo y disponer de profesionales preparados capaces de visualmente inspeccionar los cultivos en busca de síntomas.

El objetivo de este reto es entrenar un modelo robusto de regresión de imágenes, que procese imágenes de parcelas de campo individuales y prediga su valor de evaluación. Este no es un problema trivial para los métodos de análisis convencionales, debido a la alta complejidad y dimensionalidad de la información visual de las imágenes.

Nosotros nos hemos decantado por un modelo basado en una red neural convolucional, desarrollado utilizando la librería Pytorch (concretamente la librería Lightning), para analizar las imágenes y poder así predecir las enfermedades de los cultivos.

1.2 Gestión de datos y preprocesamiento

Se dispone de imágenes multiespectrales, es decir, no corresponden a imágenes RGB, sino que las imágenes tienen 5 o 10 bandas distintas. Además, las imágenes tienen tamaños distintos. Existen tres posibles enfermedades observadas en los cultivos, por lo que en una tabla se ofrece el porcentaje de cada enfermedad en cada parcela. A continuación hemos detallado una serie de desafíos con los que nos hemos encontrado a la hora de gestionar y tratar los datos para el modelo.

Formato de las etiquetas: Puesto que solo se tiene acceso a los valores de los porcentajes de cada enfermedad, sin ningún tipo de información espacial sobre su distribución, se ha descartado el utilizar modelos de segmentación semántica de imágenes y se ha tratado el problema directamente como una tarea de regresión.

Tamaño de los datos: El set de datos completo ocupa tanta memoria (~11 GB) que no nos ha sido posible cargarlo al completo y entrenar el modelo de manera simultánea en nuestros portátiles. Para solventar la situación, decidimos cargar las imágenes de manera dinámica, accediendo a cada archivo según fuera necesario por medio de los cargadores de datos de la librería Pytorch Lightning.

Regularización de las imágenes: Como se ha mencionado, las imágenes tienen un tamaño y número de canales variable, esto dificulta el proceso de regresión, ya que los modelos necesitan una entrada de datos con estructura uniforme. Por lo tanto, se han utilizado solamente las 5 bandas comunes a todas las imágenes. Además, todas las imágenes han sido redimensionadas a tamaño 330x110, este tamaño ha sido elegido por ser la media de tamaños de todas las imágenes.

Normalización de las imágenes: A la hora de utilizar redes neuronales, es recomendable estandarizar (media nula, desviación estándar 1) los datos de entrada antes de introducirlos al modelo, con el fin de estabilizar los algoritmos de entrenamiento. De nuevo, ya que es complicado trabajar con el dataset completo, hemos utilizado un algoritmo especial para calcular la media y desviación estándar de los datos por canal sin tener que cargar los datos al completo en memoria. Estos valores, necesarios para el proceso de inferencia, se guardan en un fichero adicional para evitar tener que recalcularlos. Solo es necesario computarlos cuando se cambia el set de datos de entrenamiento.

Aumentado de los datos: Para aprovechar al máximo los datos disponibles y evitar un sobre ajuste a los datos de entrenamiento, se han aplicado transformaciones aleatorias a los datos utilizados durante el proceso de entrenamiento. Solamente se han aplicado reflexiones verticales y horizontales, puesto que otro tipo de transformaciones (recorte, rotación) no conservan el área de píxeles en la imagen, lo cual podría dañar el rendimiento del modelo al tratarse de una tarea de regresión simple.

Datos adicionales: Dado que los nombres de los archivos de las imágenes contienen información variada que puede ser relevante a la hora de hacer regresión, se ha decidido utilizar esta información en la regresión para evaluar si mejoran los resultados. Las variables adicionales extraídas son:

- **Mes:** Puede ser una condición determinante, ya que algunas enfermedades pueden ser más frecuentes que otras en diferentes estaciones.
- **Cámara/dron:** Aunque en principio los instrumentos están calibrados y comparten las bandas utilizadas, al tratarse de dispositivos físicos podrían tener distintas sensibilidades.

1.3 Estrategia de particionado de datos

A la hora de comprobar los modelos, las imágenes etiquetadas se han dividido en conjuntos de entrenamiento (70%, 5968 muestras), test (15%, 1279 muestras) y validación (15%, 1279) de manera aleatoria. No hemos tenido en cuenta ninguna estrategia de estratificación, puesto que no disponíamos de métrica clara para dicho propósito. Para entrenar el modelo final "Normal+" - el utilizado para realizar la predicción de la competición - se han dividido en 90% entrenamiento, 10% evaluación y 0% test para maximizar la cantidad de información utilizada.

2 Desarrollo

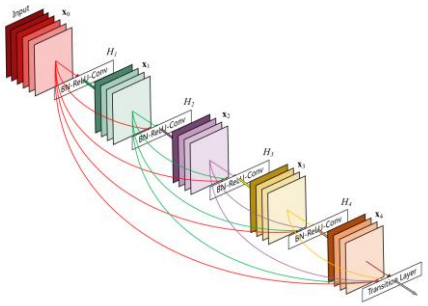
2.1 Descripción del algoritmo

Se ha utilizado una red neuronal convolucional de cara a resolver el problema, debido al tipo de datos de entrada. En concreto, nos hemos basado en la arquitectura DenseNet a la hora de diseñar el modelo de la red. Existen arquitecturas no-convolucionales más recientes para el tratamiento de imágenes basadas en Transformers, pero su principal ventaja yace en explotar información

global de las imágenes, lo que principalmente es de utilidad en tareas de clasificación. Sin embargo, esa información global no es necesaria para tareas de regresión por píxeles como esta.

En una DenseNet, cada una de las capas obtiene la salida de las capas precedentes y pasa sus datos de salida a todas las subsecuentes (imagen). Esta arquitectura alivia el desvanecimiento del gradiente, fortalece la propagación de características o *features* incitando a su reutilización y reduce sustancialmente el número de parámetros necesarios respecto a arquitecturas más sencillas como ResNet.

A la hora de diseñar el modelo nos hemos basado en la arquitectura densenet-121 que presentan en el artículo original. En esencia, en lo referente al *backbone* de la red tan solo hemos modificado el número de canales de entrada para adaptarlos a las imágenes multiespectrales (de 3 a 5) y reducido el ratio de crecimiento con el fin de reducir el tamaño del modelo (de 64 a 12, puesto que presentaban un rendimiento similar con menos parámetros).



También hemos añadido un decoder personalizado al final de la red, que básicamente se basa en capas densas. Las dos primeras incluyen la salida del bloque convolucional y múltiples señales correspondientes al mes del año y al sensor (dron) utilizado para obtener las imágenes. La última solo tiene tres neuronas con una función de activación sigmoide, que representan los resultados de la predicción (porcentajes de cada enfermedad, entre 0 y 1). La inclusión de los datos adicionales la hemos programado como opcional, de manera que hemos hecho pruebas con ambas versiones (con datos extra "Extra" y sin datos extra "Normal").

2.2 Entrenamiento

Nuestra estrategia ha estado principalmente limitada por los recursos computacionales de los que disponíamos.

Hiper Parámetros: Como se explica en la sección anterior, no hemos optimizado los hiper parámetros de la red, simplemente hemos tomado la arquitectura presentada en el artículo original, adaptándola a los datos y reduciendo un poco su tamaño.

Batch Size: Se utilizó un tamaño batch de 48 para todas las pruebas, ya que era el máximo posible en nuestros equipos debido a limitaciones de memoria en la GPU.

Learning rate: Utilizamos un regulador del ratio de aprendizaje, de manera que fuese descendiendo (de 10^{-5} a 10^{-7}) según avanzaba la optimización. Esto se hace para facilitar la convergencia de la red y evitar oscilaciones en torno al mínimo local.

Convergencia: La condición de parada se estableció en minimizar el MAE del set de validación de los datos. Si pasadas 5 épocas de entrenamiento no se mejora dicha métrica, se para el entrenamiento y esa guarda la mejor versión de los pesos.

Rendimiento: En términos de hardware, el entrenamiento se ha llevado a cabo en un portátil con Ubuntu 22.1 (CPU AMD Ryzen 7, GPU NVIDIA RTX 3050). El modelo final cuenta con 1,1M de parámetros, resultando en un tiempo de entrenamiento de algo menos de 1 hora y en tiempos de inferencia de unos 40ms en nuestra plataforma.

3 Resultados

En la Tabla siguiente se pueden ver los resultados de los entrenamientos para las dos variantes del modelo. Es importante destacar que en los modelos "Normal" y "Extra" se han dado utilizando solo un 70% de los datos en el proceso de entrenamiento como tal, y, por lo tanto, el rendimiento final en producción puede ser algo mayor. Las predicciones del fichero CSV entregado se hicieron con el modelo "Normal +" entrenado con un split más agresivo (90% train, 10%eval, 0% test) de los datos.

Modelo	Validation		Test		Épocas	Tiempo
	MAE	R^2	MAE	R^2		
Normal	2,22	0.8581	2,71	0.8239	84	42
Normal +	1,98	0.8529	-	-	18 (+ 84)	12
Extra	2,30	0.8218	2,33	0.7866	88	54

En las Figuras 1 y 2 se muestran las predicciones del set de evaluación del modelo "Normal +". En la FIgura 1, la situación ideal se daría si todos los puntos estuviesen en la diagonal (regresión perfecta). En la Figura 2 se puede observar como las distribuciones de las distintas enfermedades en el mismo set. En ambos casos se han omitido los valores 0 y 1, que representan un alto porcentaje de las muestra del dataset, y una de las mayores fuentes de ruido del modelo (al redondear entre 0 y 1 falla bastante).

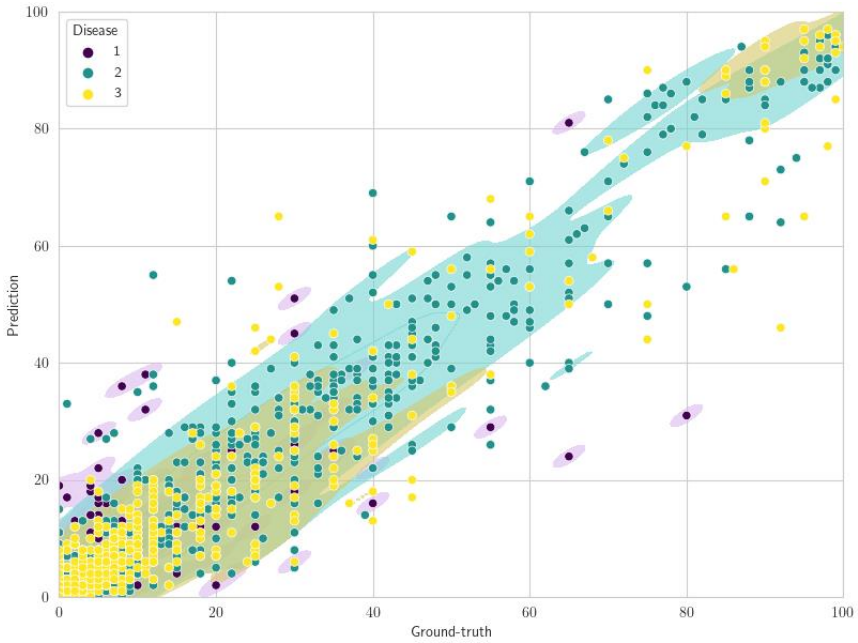


FIgura 1: Representación de las predicciones del modelo frente a los valores reales en el test de validación del modelo "Normal +".

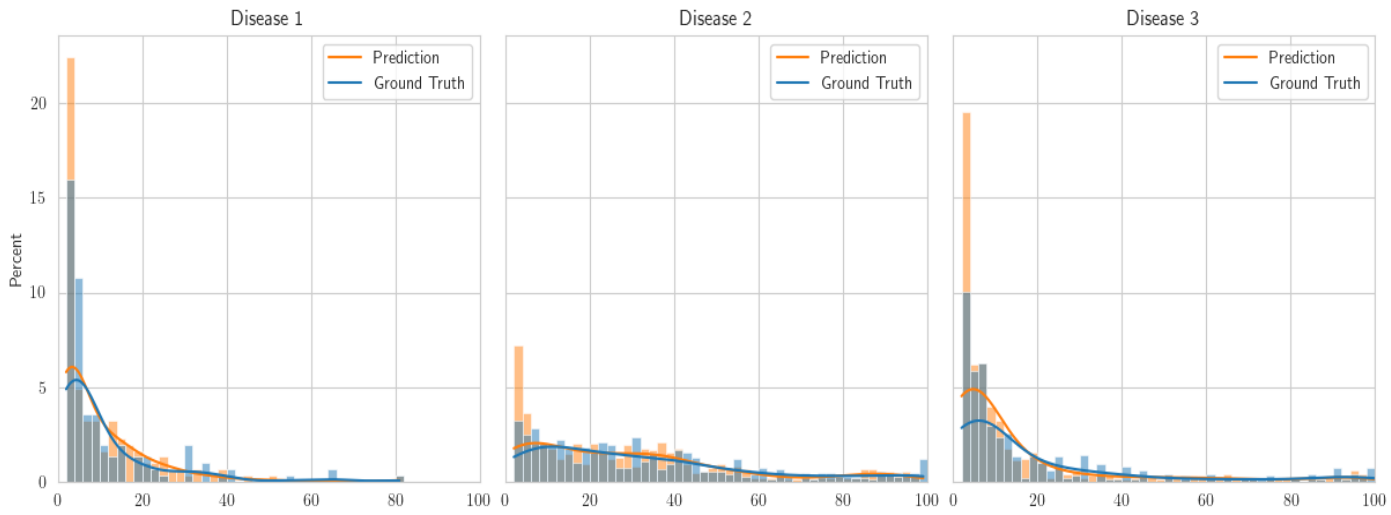


FIgura 2: Distribuciones de las enfermedades en el test de validación del modelo "Norma +".

4 Conclusiones

En resumen, hemos tratado el problema propuesto como una regresión, en la que hemos intentado predecir las proporciones de cada enfermedad de una parcela de cultivo dada la imagen. Para la regresión, hemos hecho uso la red convolucional DenseNet, la cual es una de las que ha demostrado mayor rendimiento. También hemos probado en añadir variables extra accesibles en los ficheros, los cuales podían aportar información determinante para mejorar la regresión, aunque como se ha visto no ha sido el caso. Esto puede haber sucedido porque estas variables condicionan demasiado la salida de la red neuronal, por lo que puede que por ello se haya perdido generalización.

Por otro lado, creemos que uno de los factores más limitadores para nuestro modelo ha sido los pocos casos de la enfermedad 1. Esto se puede apreciar sobre todo en la Figura 1, donde la enfermedad 1 tiene outliers más pronunciados que el resto.

En términos de escalabilidad nuestro modelo consta de alrededor de un millón de parámetros, lo cual entra dentro de lo estándar para una red neuronal. Además, como se ha mencionado anteriormente, se ha requerido de tan solo una hora para su entrenamiento. Con mayores recursos se podría entrenar más rápido, ajustar los hiper parámetros o aumentar el tamaño de batch. También se podría aumentar el tamaño del modelo, aunque creemos que este no es un factor limitante en su rendimiento - sobre todo comparado con la imprecisión de las propias etiquetas.

Además, el modelo destaca por su facilidad de implementación, ya que se trata simplemente de un módulo de PyTorch, fácilmente exportable a cualquier entorno de producción si se dispone de los ficheros con los pesos y las constantes de normalización de los datos.