Proyecto 3
Aprendizaje no supervisado

1st Camacho Hernández José Julián
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
jcamacho341@estudiantec.cr

2nd Guillén Fernández José Leonardo
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
leoguillen@estudiantec.cr

3rd Guzmán Quesada Joshua
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
joshagq@estudiantec.cr

***Resumen*—This paper will explain and try to prove the validy of two hipothesys using the Plant Village Dataset. The first one states that it is possible to train an autoencoder by reconstructing unlabeled data, and by transfer learning train a classifier, and achieve better results than by solely using labeled data directly to train a classifier. On the other hand, the second one states that by adding noise to the input of the autoencoder, it is possible to learn more robust representations that improve classification. To prove these, we will train some Keras neural networks models to get predictions of whether a plant is healthy or not. The results will be compared by computing some metrics like accuracy, recall or F1 score. The results showed that the validity of hypothesis 1 could be verified.**

***Index Terms*—Machine Learning, Plant deseases, Autoencoders, Convolutional Networks (CNN), Denoising**

I. Introducción

En el campo de la agricultura, la salud de las plantas es un factor crucial para garantizar una buena cosecha y prevenir pérdidas significativas. Tradicionalmente, los agricultores han confiado en la experiencia visual para detectar enfermedades en las plantas, lo que puede ser un proceso lento y propenso a errores. Sin embargo, en los últimos años, el desarrollo de técnicas de aprendizaje automático ha abierto nuevas posibilidades para la detección temprana y precisa de enfermedades vegetales.

Uno de los enfoques prometedores en este campo es el uso de *autoencoders* y modelos de clasificación. Los *autoencoders* son una forma de red neuronal que se utiliza para aprender representaciones latentes de los datos de entrada sin supervisión. En el caso de las plantas, los *autoencoders* pueden aprender características relevantes de las imágenes de las hojas o tallos enfermos y no enfermos, capturando patrones sutiles que pueden no ser fácilmente discernibles para el ojo humano.

Una vez que se ha entrenado un *autoencoder* para aprender estas representaciones latentes, se puede utilizar un modelo de clasificación para determinar si una planta está enferma o no. El modelo de clasificación puede basarse en diferentes algoritmos, como redes neuronales convolucionales (CNN) o perceptrones multicapa (MLP), que utilizan las características extraídas por el *autoencoder* para tomar decisiones.

El uso de *autoencoders* y modelos de clasificación para la detección de enfermedades en plantas ofrece varias ventajas. En primer lugar, permite una detección temprana y precisa, lo que puede ayudar a los agricultores a tomar medidas rápidas y específicas para controlar la propagación de enfermedades y minimizar las pérdidas de cultivos. Además, al automatizar el proceso de detección, se reduce la dependencia de la experiencia humana y se evitan posibles sesgos o errores de interpretación.

Varios estudios han demostrado la eficacia de esta metodología en la detección de enfermedades vegetales. Por ejemplo, en un estudio realizado por Mohanty et al. (2016), se utilizó un *autoencoder* junto con una red neuronal convolucional para detectar enfermedades en hojas de arroz. Los resultados mostraron altas tasas de precisión en la clasificación de hojas sanas y enfermas. [1]

El presente proyecto consiste en la utilización del set de datos *Plant Village* o *Plant leave diseases dataset* [2] para verificar la veracidad de dos hipótesis relacionadas con predicciones usando *autoencoders*. Estas son:

1. En ausencia de datos etiquetados, es posible entrenar un *autoencoder* reconstruyendo datos sin *labels*, haciendo *transfer learning* con el subconjunto pequeño que sí tiene *labels*, y obtener mejores resultados que con solo usar el mismo subconjunto pequeño con *labels* directo para entrenar un clasificador.
2. Al agregar ruido a la entrada del *autoencoder*, es posible aprender representaciones más robustas que mejoran la clasificación y por ende, la representación latente

En la siguiente sección del presente documento, se detallará el diseño de los autoencoders y clasificadores utilizados para el análisis de las dos hipótesis plantedas. Seguidamente, en la sección III se analizarán los resultados obtenidos al aplicar diversas técnicas como la congelación de pesos en el *autoencoder* o el uso de ruido en el conjunto de datos. Finalmente, se discutirán las conclusiones del proyecto.

II. Detalles del diseño del programa desarrollado

En la presente sección se explica el diseño de las redes neuronales implementadas.

II-A. *Experimento 1*

II-A1. Clasificador sin autoencoder: En este primer experimento, en primer lugar se utiliza un clasificador para realizar entrenamiento y predicciones desde cero. Para esto de usará una red convolucional. La siguiente figura se presenta el diagrama de flujo del diseño de una red convolucional.

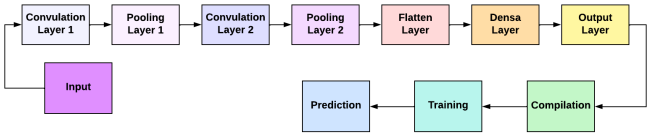

Figura 1. Diagrama de Flujo de una Red Convolucional.

A continuación se detalla cada una de las partes que lo componen:

- **Entrada**: Esta etapa representa las imágenes del set de datos *Plant Village* que se utilizarán como datos de entrada para el modelo de CNN.
- **Capa Conv2D**: Es una capa convolucional que realiza operaciones de convolución en las imágenes de entrada. Cada filtro en esta capa extrae características específicas de la imagen, y el número de filtros se determina mediante el parámetro "Número de filtros (X)". El tamaño del kernel determina el tamaño de la ventana de convolución que se desliza sobre la imagen. La función de activación ReLU se aplica a los mapas de características resultantes para introducir no linealidad. El relleno "same."segura que el tamaño de salida sea el mismo que el de la imagen de entrada.
- **Capa Pooling**: Es una capa de submuestreo que reduce la dimensión espacial de los mapas de características obtenidos de la capa convolucional. El tamaño de pool (Z) define el tamaño de la ventana de agrupación que se desliza sobre la imagen y selecciona el valor máximo o promedio de cada región, dependiendo del tipo de pooling utilizado.
- **Repetir capas convolucionales y de pooling:** Estas capas se repiten según sea necesario para construir la arquitectura de la CNN. Añadir más capas convolucionales y de pooling puede ayudar a extraer características más complejas y abstractas de las imágenes.
- **Capa Flatten**: Esta capa se utiliza para aplanar la salida de la última capa convolucional, lo que significa convertir la salida en un vector unidimensional. Prepara los datos para ser alimentados en una capa densa.
- **Capa Densa**: Es una capa completamente conectada en la que cada neurona está conectada a todas las neuronas de la capa anterior. El número de neuronas se determina mediante el parámetro "Número de neuronas (N)". La función de activación ReLU se aplica a las salidas de las neuronas.
- **Capa de Salida:** Es la capa final del modelo, que produce la salida deseada. En este caso, hay una sola neurona en la capa de salida y la función de activación utilizada es la sigmoide. Esto se debe a que se trata de una clasificación binaria determinando si la planta está enferma o no.
- **Compilación:** Esta etapa configura el modelo para el entrenamiento, especificando la función de pérdida y el optimizador. La función de pérdida mide qué tan bien el modelo se ajusta a los datos de entrenamiento, y el optimizador ajusta los pesos del modelo para minimizar la pérdida.
- **Training:** Aquí es donde el modelo se entrena utilizando los datos de entrenamiento. Durante el entrenamiento, el modelo ajusta sus parámetros iterativamente para mejorar su rendimiento en la tarea de clasificación.
- **Predicción:** Después del entrenamiento, el modelo se evalúa utilizando los datos de prueba para medir su rendimiento en datos no vistos previamente. Esto proporciona una evaluación objetiva de la capacidad del modelo para generalizar y clasificar correctamente nuevas imágenes de plantas.

Basado en la explicación del diagrama de flujo de las redes convolucionales, se define la estructura de los clasificadores CNN utilizados en el proyecto como se presenta en la figura 2 a continuación.

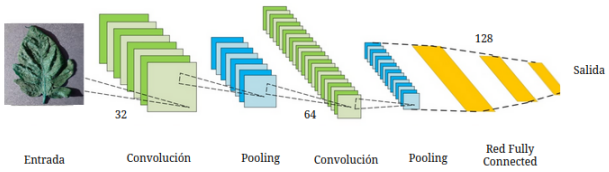

Figura 2. Diagrama de arquitectura de las CNN utilizadas.

Se observa que se cuenta con dos capas de convolución, así como dos etapas de *pooling*. Finalmente, se pasa a una red completamente conectada que consta de una capa oculta de 128 neuronas y una capa de salida de 2, ya que se trata de clasificación binaria.

II-A2. Clasificador con pesos del autoencoder congelados: Seguidamente, se utiliza un clasificador de red convolucional igual al descrito en la sección anterior, pero cuya entrada es el espacio latente obtenido de un *autoencoder* con pesos congelados.

Un *autoencoder* es un tipo de red neuronal que se utiliza en el aprendizaje automático para la compresión y reconstrucción de datos. Se considera un modelo auto-supervisado, lo que significa que no requiere etiquetas externas para el entrenamiento, sino que utiliza los datos de entrada como su propia "supervisión".

La estructura básica de un *autoencoder* consta de dos partes principales: la capa de codificación (encoder) y la capa de decodificación (decoder). El *encoder* transforma los datos de entrada en una representación de menor dimensionalidad, llamada *espacio latente*. Esta representación captura las características esenciales de los datos y contiene una versión comprimida de la información original. Luego, el *decoder* toma esta representación latente y la reconstruye para producir una versión aproximada de los datos de entrada originales.

En la figura 3, se presenta un diagrama que ejemplifica la arquitectura de un *autoencoder* compuesto por dos redes convolucionales como las que se explicaron en la sección anterior.

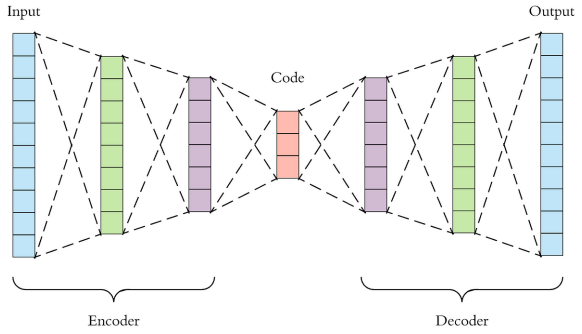

Figura 3. Diagrama de un *autoencoder*

El objetivo principal de un *autoencoder* es minimizar el error de reconstrucción entre los datos de entrada y la versión reconstruida. Durante el entrenamiento, se ajustan los pesos y los sesgos de las capas del *encoder* y del *decoder* para mejorar la precisión de la reconstrucción.

Seguidamente, en la figura 4 se presenta el digrama de flujo de un *autoencoder*.

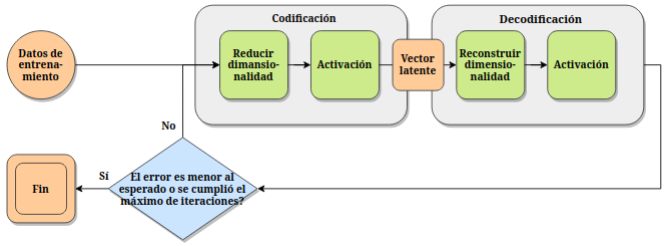

Figura 4. Diagrama de flujo de un *autoencoder*

En la figura 6, se visualiza la estructura que se implementa para realizar el *transfer learning* utilizando el vector latente del *autoencoder* con los pesos congelados como entrada para la CNN.

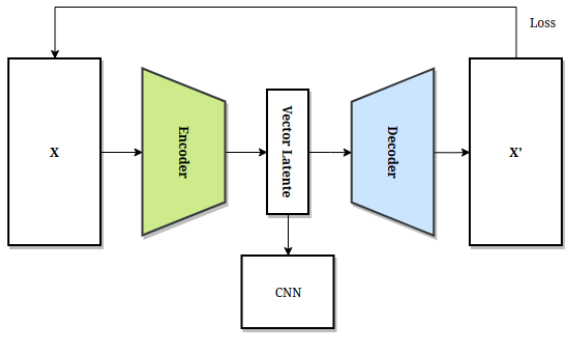

Figura 5. Diagrama de la implementación del *autoencoder* con el modelo de CNN

II-A3. Clasificador con pesos del autoencoder sin congelar: Adicionalmente, se implementa un clasificador CNN como el explicado en la sección II-A1 junto con un *autoencoder*, resultando en una estructura similar a la detallada en la sección anterior. Sin embargo, en este caso se establece que los pesos no se congelen. Esto causará que los pesos del *autoencoder* se irán ajustando durante el entrenamiento del modelo clasificador.

II-B. Experimento 2

El experimento 2, consiste en comparar dos clasificadores basados en *autoencoders*, con la diferencia de que uno de ellos fue entrenado agregando algún tipo de ruido a las imágenes de entrada. Este tipo de modelo se conoce como *Denoising Autoencoder*. Estos modelos serán entrenados con un conjunto de datos previamente utilizado en el Experimento 1.

En el caso específico de los *Denoising Autoencoders*, como se mencionó, se introducen perturbaciones o ruido en las imágenes de entrada durante el entrenamiento. Esto ayuda al *autoencoder* a aprender a reconstruir los datos originales a pesar del ruido, lo que puede resultar en una representación latente más robusta y generalizable.

Para evaluar la calidad de las representaciones latentes aprendidas por los *autoencoders*, se realizará una visualización de los vectores latentes de las imágenes sin etiquetas. Estos vectores latentes serán visualizados utilizando técnicas de reducción de dimensionalidad como PCA (Análisis de Componentes Principales). La idea es representar los vectores latentes en un espacio de menor dimensión, donde se puedan identificar patrones o clústers correspondientes a las diferentes clases de objetos.

La visualización de los vectores latentes permitirá determinar si los modelos han aprendido a modelar las clases de manera efectiva. Se espera que el *Denoising Autoencoder* muestre clústers más claros y distintos para las diferentes clases, lo que indicaría una mejor separación de las representaciones latentes en comparación con el *autoencoder* convencional.

La visualización de los vectores latentes y la comparación de los clasificadores basados en *autoencoders* proporcionará información sobre la capacidad de los modelos para aprender características relevantes y discriminar entre diferentes clases de objetos. Esto puede ser útil para evaluar la efectividad de los *autoencoders* en tareas de clasificación y comprender la calidad de las representaciones latentes aprendidas.

II-B1. Autoencoder Normal: Este *autoencoder* tendrá la misma arquitectura que el presentado en secciones anteriores. Para este caso, el modelo se compila utilizando el optimizador Adam y la función de pérdida de error cuadrático medio (MSE) para que el decodificador pueda aprender a reconstruir las imágenes originales.

El Autoencoder se entrena utilizando los datos de entrenamiento, pasando la misma entrada como objetivo de salida para que el modelo pueda aprender a reconstruir las imágenes sin ruido, tal como se presenta en el siguiente diagrama de flujo:

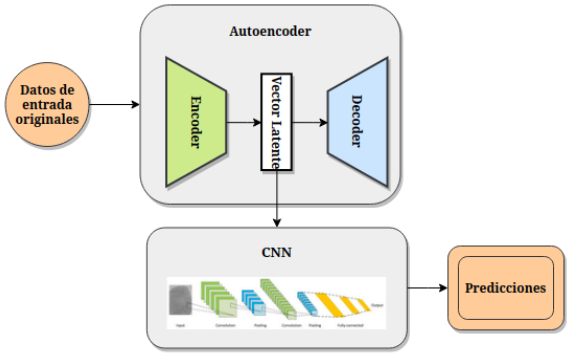

Figura 6. Diagrama de flujo de la implementación del experimento 2.

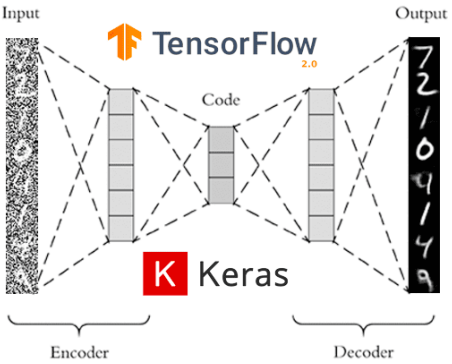

Figura 7. Diagrama de la implementación del *Denoising Autoencoder*

Una vez entrenado el Autoencoder, se obtienen los vectores latentes de las imágenes de prueba utilizando el codificador. Estos vectores latentes se aplanan y luego se aplica PCA para reducir su dimensionalidad a 2D.

Luego, se visualizan los vectores latentes en un gráfico de dispersión utilizando las dos componentes principales obtenidas mediante PCA. Cada punto en el gráfico representa una imagen de prueba, y se utiliza el color para mostrar las etiquetas conocidas (en este caso, se supone que las etiquetas están disponibles aunque no se usaron en el entrenamiento).

Después de la visualización, se construye un modelo clasificador utilizando la salida del decodificador del *autoencoder* como entrada. El clasificador consta de capas adicionales (aplanamiento, capa densa con activación relu y una capa densa de salida con activación sigmoidal) para realizar la clasificación binaria.

Finalmente, el modelo clasificador se entrena utilizando los datos de entrenamiento originales (no los datos reconstruidos por el *autoencoder*) y se evalúa utilizando los datos de prueba.

II-B2. Denoising Autoencoder: Basado en el modelo anterior, la diferencia es que se introducen imágenes con ruido como datos de entrada durante el entrenamiento del *autoencoder*. Esto se logra generando imágenes ruidosas al agregar un factor de ruido a las imágenes originales. Luego, se utiliza el conjunto de imágenes ruidosas como entrada y el conjunto de imágenes originales como objetivo de salida para entrenar el modelo.

Después de entrenar el *autoencoder* con imágenes ruidosas, se procede de manera similar al modelo anterior. Se obtienen los vectores latentes de las imágenes de prueba utilizando el codificador y se aplican técnicas de reducción de dimensionalidad, como PCA, para visualizar los vectores latentes en un gráfico de dispersión en 2D.

La construcción del modelo clasificador basado en el *autoencoder* y su entrenamiento se realiza de manera idéntica al modelo anterior. El clasificador utiliza la salida del decodificador como entrada y se compila con los mismos parámetros, optimizador y función de pérdida. El objetivo del clasificador es utilizar las características extraídas por el *autoencoder* para realizar la clasificación binaria de las imágenes.

Finalmente, se evalúa el rendimiento del modelo clasificador utilizando los datos de prueba y se calculan métricas de evaluación, como la precisión, para determinar qué tan bien el modelo es capaz de realizar clasificaciones precisas en función de las imágenes de entrada.

En la figura 7 se presenta el diagrama que ejemplifica la implementación de un *denoising autoencoder*.

III. Resultados y análisis

En la presente sección, se detallan aspectos relacionados con los resultados obtenidos al llevar a cabo el ciclo de diseño de un problema de reconocimiento de patrones.

III-A. Experimento 1

El conjunto de datos utilizado en este estudio se divide en dos partes: datos sin etiquetas para el *encoder* y datos con etiquetas para el clasificador.

III-A1. Ejecución 1: En esta primera ejecución, se define que el 80 % de los datos sin etiquetas se utiliza para entrenar el *autoencoder*, mientras que el 20 % restante se utiliza para evaluar el desempeño del *autoencoder* sin las etiquetas. Por otro lado, el conjunto de datos con etiquetas se divide en un 10 % para entrenamiento y un 10 % para pruebas.

III-A1a. Clasificador sin autoencoder (A): Se puede observar que el modelo basado en el *autoencoder* con datos de prueba, cuyos resultados se observan en la figura 8, tiene un desempeño significativamente mejor en comparación con el modelo basado en los datos de entrenamiento presentados en la figura 9.

El modelo con datos de prueba muestra una precisión, *recall* y F1 *Score* más altos, lo cual indica que es capaz de predecir con mayor precisión y cobertura los valores positivos y negativos de la variable objetivo. El alto valor de AUC también sugiere un buen rendimiento del modelo en términos de clasificación.

Por otro lado, el modelo con datos de entrenamiento muestra un rendimiento más bajo en todas las métricas evaluadas. Esto puede indicar que el modelo no ha generalizado bien a nuevos datos y puede estar sobreajustado (*overfitting*) a los datos de entrenamiento.

Por lo que el modelo basado en el *autoencoder* con datos de prueba muestra un mejor rendimiento y una mayor capacidad

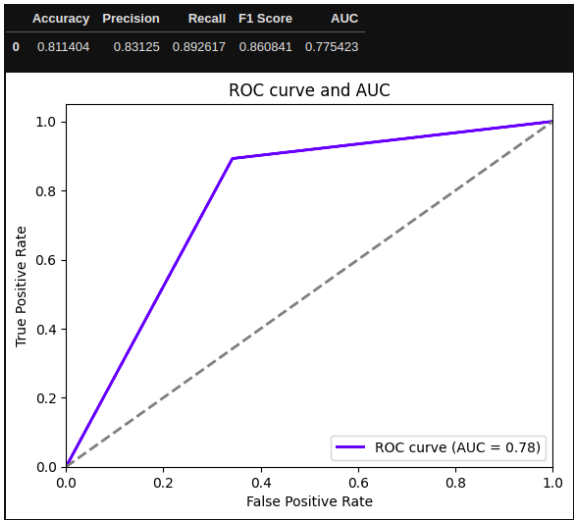

Figura 8. Resultados de métricas para el conjunto de prueba.

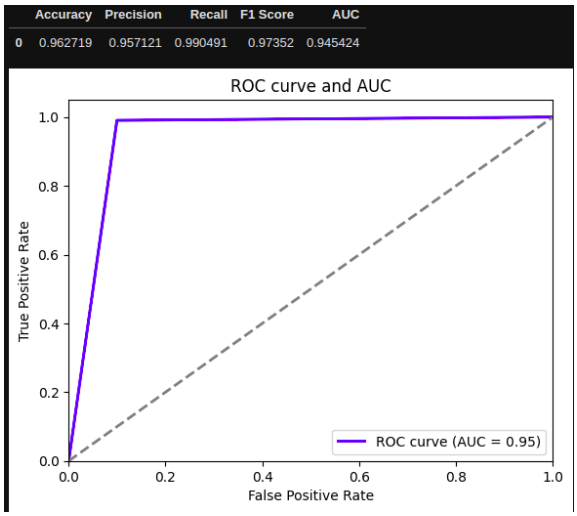

Figura 9. Resultados de métricas para el conjunto de entrenamiento.

de generalización en comparación con el modelo basado en los datos de entrenamiento. Esto destaca la importancia de evaluar el desempeño del modelo en datos no vistos durante el entrenamiento para obtener una evaluación más precisa de su capacidad de predicción.

III-A1b. Clasificador con pesos del autoencoder congelados (B): En el primer modelo 11, que utiliza los datos de prueba, se observa que la precisión, *recall* y F1 *Score* son relativamente bajos en comparación con el segundo modelo. Esto podría indicar que el modelo tiene dificultades para capturar correctamente los verdaderos positivos y negativos, y puede haber un mayor número de falsos positivos y falsos negativos en las predicciones.

En contraste, en el segundo modelo con datos de entrenamiento de la figura 11, se observa un aumento en la precisión, *recall* y F1 *Score*. Esto sugiere que el modelo ha mejorado su capacidad para predecir de manera más precisa y completa

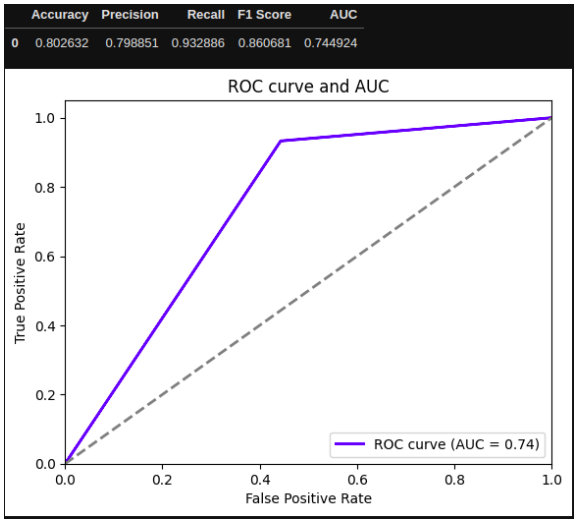

Figura 10. Resultados de métricas para el conjunto de prueba.

los valores positivos y negativos. Además, el valor de AUC también ha aumentado, lo que indica un mejor rendimiento general en términos de clasificación.

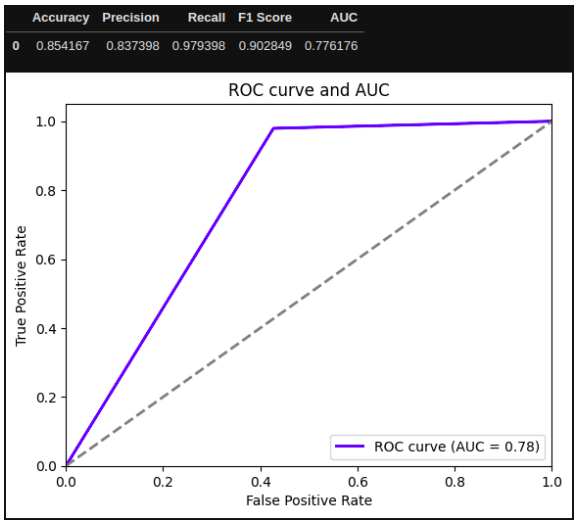

Figura 11. Resultados de métricas para el conjunto de entrenamiento.

Entonces al utilizar un *autoencoder* con pesos congelados, se observa una mejora en el rendimiento del modelo al predecir con datos de prueba en comparación con los datos de entrenamiento. Esto indica que el modelo ha generalizado mejor a nuevos datos y ha mejorado su capacidad de predicción.

III-A1c. Clasificador con pesos del autoencoder sin congelar (C): El primer modelo, que utilizó un *autoencoder* sin los pesos congelados y predijo con datos de prueba, mostró un rendimiento inferior en comparación con los modelos anteriores, tal como se visualiza en la figura 12.

El modelo tuvo dificultades para capturar correctamente los verdaderos positivos y negativos, lo que resultó en métricas más bajas de precisión, *recall* y F1 *Score*. Esto indica que la

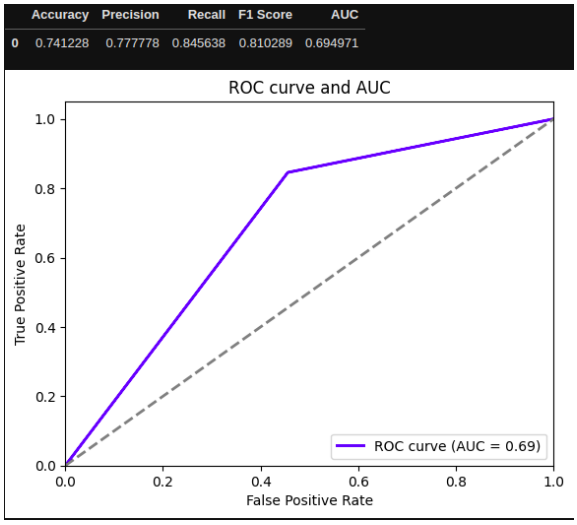

Figura 12. Resultados de métricas para el conjunto de prueba.

capacidad de predicción del modelo fue limitada y no logró clasificar con alta precisión.

En contraste, el segundo modelo, también basado en un *autoencoder* sin los pesos congelados, pero con predicciones realizadas en datos de entrenamiento 13, mostró un rendimiento considerablemente mejor. Las métricas de precisión, *recall*, F1 Score y AUC aumentaron significativamente, lo que indica una mejora en la capacidad del modelo para predecir correctamente los valores positivos y negativos y una mejor capacidad de clasificación en general.

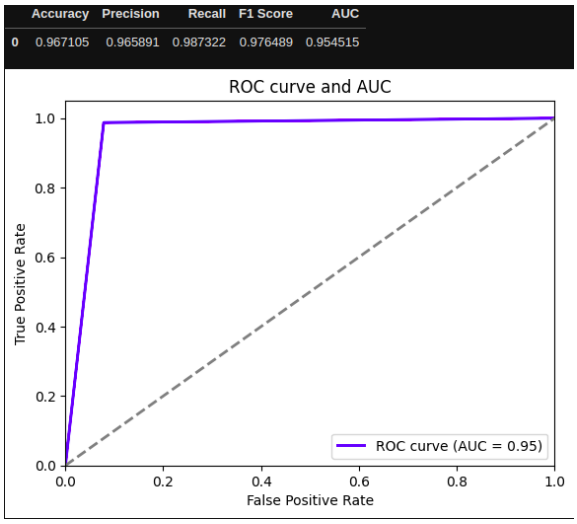

Figura 13. Resultados de métricas para el conjunto de entrenamiento.

Comparando los tres modelos anteriores, se concluye que el mejor rendimiento se obtuvo con el modelo basado en el *autoencoder* con pesos congelados y prediciones realizadas en datos de prueba. Este modelo logró la mayor precisión, *recall*, F1 *Score* y AUC, demostrando una predicción más precisa y una mejor capacidad de clasificación en general.

Es importante tener en cuenta que estos resultados son específicos para el conjunto de datos y las configuraciones utilizadas en este análisis. Dependiendo del problema y los datos particulares, los resultados pueden variar. Por lo tanto, se recomienda realizar más experimentos y evaluaciones para determinar el mejor modelo en cada caso específico.

III-A2. Ejecución 2: En la segunda ejecución se mantienen los modelos pero se varían los porcentajes de división del set de datos.

III-A2a. Clasificador sin autoencoder (D): El uso de un 50 % de datos sin etiquetas en el entrenamiento del encoder implica que el modelo se está entrenando en una configuración de aprendizaje no supervisado. Esto puede ser beneficioso para aprender características latentes o representaciones intrínsecas de las imágenes de plantas sin depender de las etiquetas. Sin embargo, esto también significa que solo se están utilizando la mitad de los datos con etiquetas disponibles para entrenar y evaluar el clasificador.

Al asignar un 35 % de datos con etiquetas para el entrenamiento, estás limitando la cantidad de muestras etiquetadas utilizadas para ajustar los parámetros del clasificador. Esto puede llevar a una menor capacidad del modelo para aprender patrones discriminativos y, por lo tanto, afectar su rendimiento.

Por último, el 15 % de datos con etiquetas destinados a las pruebas permite evaluar el rendimiento del clasificador en un conjunto de datos independiente. Sin embargo, dado que la cantidad de datos de prueba es relativamente pequeña, es posible que la evaluación no sea completamente representativa del rendimiento real del modelo.

En cuanto al resultado de *recall* del 0.949416, tal como se aprecia en la figura del clasificador 14que indica una tasa alta de detección de plantas infectadas con hongos, esto es positivo en el contexto de la detección de enfermedades en plantas.

Un alto *recall* implica que el modelo tiene una capacidad considerable para identificar las muestras positivas reales. Esto es especialmente relevante en aplicaciones agrícolas, donde es crucial detectar las plantas enfermas para tomar medidas de control o tratamiento adecuadas.

El hecho de no utilizar *autoencoders* en este escenario implica que el modelo se basa principalmente en los datos etiquetados disponibles para aprender a clasificar las plantas. Si bien los *autoencoders* pueden ser útiles para extraer características relevantes de las imágenes y mejorar la capacidad de generalización del modelo, en este caso, el clasificador ha logrado un buen rendimiento en términos de *recall* incluso sin el uso de *autoencoders*.

III-A2b. Clasificador con pesos del autoencoder congelados (E): En primer lugar, los porcentajes de datos indican cómo se dividen los conjuntos de datos para el entrenamiento y la evaluación del modelo. Aquí, se utilizó un 50 % de datos sin etiquetas para el entrenamiento del *autoencoder*, lo cual significa que estamos aprovechando imágenes sin etiquetas para aprender características relevantes de las plantas.

Luego, se destinó un 35 % de datos con etiquetas para el entrenamiento del clasificador, lo que limita la cantidad de datos con los que el modelo puede aprender a clasificar

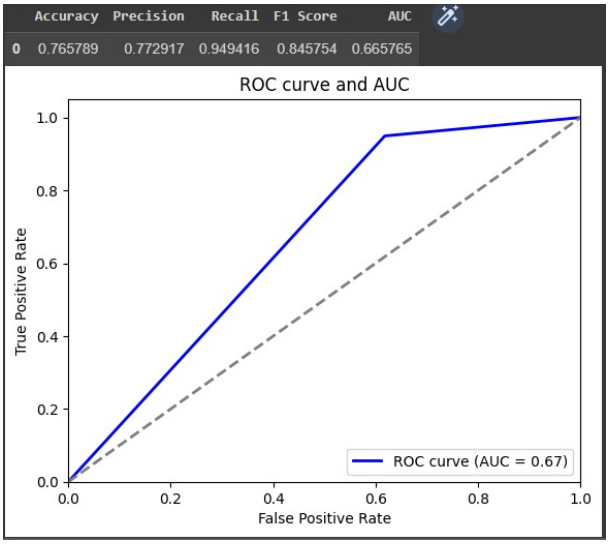

Figura 14. Clasificador sin autoencoder con un 50 % de datos sin labels, un 35 % para training y un 15 % para testing.

correctamente. Por último, el 15 % de datos con etiquetas se utilizó para evaluar el rendimiento del clasificador, lo que da como la mejor implementación al tener resultados mas certeros y precisos.

Centrándose principalmente en el resultado de *recall*, que es de 0.972763, nos indica qué tan bien el clasificador puede detectar correctamente las plantas infectadas con hongos. Un alto *recall* es importante en este contexto, ya que nos interesa identificar de manera precisa y sensible las plantas enfermas para aplicar tratamientos adecuados, tal como se puede ver en la gráfica con los resultados de este 15.

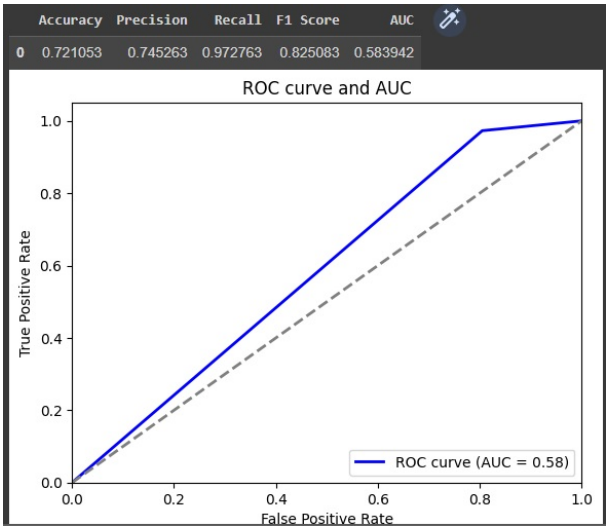

Figura 15. Clasificador con autoencoder y pesos congelados con un 50 % de datos sin labels, un 35 % para training y un 15 % para testing

En cuanto al uso del *autoencoder* con pesos congelados, significa que se aprovecha el *autoencoder* para extraer características relevantes de las imágenes, pero los pesos del *autoencoder* no se actualizan durante el entrenamiento del clasificador. Esto podría limitar la capacidad del modelo para aprender características más específicas para la tarea de clasificación, lo que podría explicar las métricas moderadas observadas.

III-A2c. Clasificador con pesos del autoencoder sin congelar (F): Dando un énfasis en el resultado de *recall*, que es de 0.949416, nos indica qué tan bien el clasificador puede detectar correctamente las plantas infectadas con hongos, como se aprecia en la figura 16. Un alto *recall* es importante en este contexto, ya que nos interesa identificar de manera precisa y sensible las plantas enfermas para aplicar tratamientos adecuados.

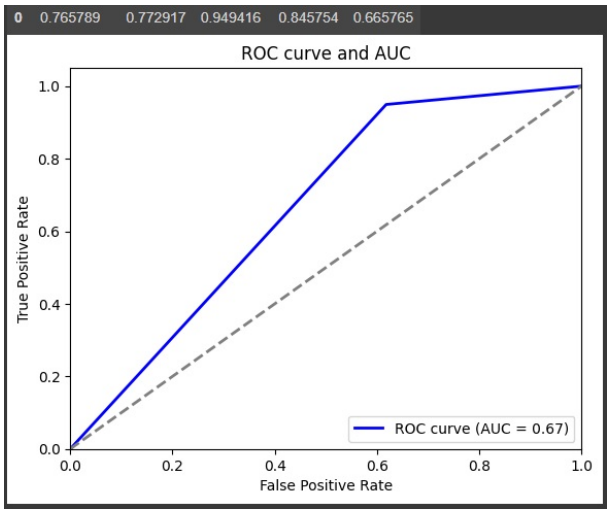

Figura 16. Clasificador con autoencoder y sin pesos congelados con un 50 % de datos sin labels, un 35 % para training y un 15 % para testing

El hecho de no utilizar un *autoencoder* implica que el modelo se basa principalmente en los datos etiquetados disponibles para aprender a clasificar las plantas. Aunque el uso de un *autoencoder* puede proporcionar ventajas adicionales al extraer características relevantes de las imágenes, en este caso, el clasificador ha logrado un buen rendimiento en términos de *recall* incluso sin el uso de un *autoencoder*.

El clasificador con *autoencoder* y pesos sin congelar muestra un buen *recall* en la detección de plantas enfermas, pero con una precisión relativamente baja. Esto significa que el modelo puede tener algunos falsos positivos. Aunque no se utilizó un *autoencoder* en este caso, el clasificador ha logrado un buen rendimiento en términos de *recall*. Sin embargo, es importante tener en cuenta los porcentajes de datos utilizados y las limitaciones asociadas con la falta de uso de un *autoencoder* en la extracción de características.

III-B. Experimento 2

Este segundo experimento consiste en comparar dos clasificadores basados en *autoencoders*, con la diferencia de que uno de ellos fue entrenado agregando el tipo de ruido *salt and pepper noise* a las imágenes de entrada.

Adicionalmente, se realiza la visualización de los vectores latentes de las imágenes ”sin *labels*”, mostrando los *labels* con colores para visualizar si en el espacio latente el modelo aprendió a modelar bien las clases, es decir, que puntos de la misma clase se encuentran cerca.

III-B1. Autoencoder normal: Para el *autoencoder* normal, en la figura 17 a continuación, se presenta el resultado de una reducción de dimensionalidad para la visualización de los vectores lantentes, utilizando PCA.

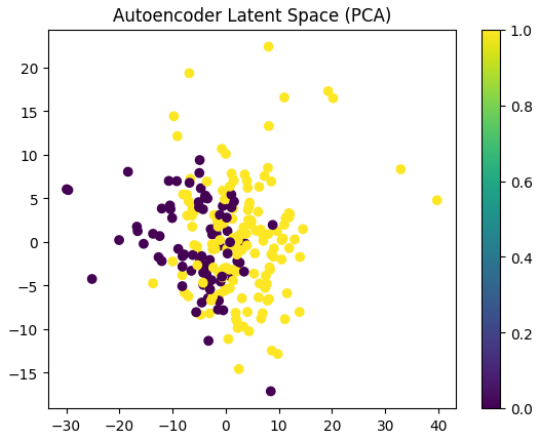

Figura 17. Representación del PCA para el autoencoder normal.

En esta se puede observar que hay dos colores de puntos, que representan los *labels* relacionados con la sanidad de la planta, que se reflejan en morado, o la enfermedad de la planta, representados por el color amarillo.

Se puede analizar que se van formando clústers, indicando que el modelo es capaz de aprender progresivamente para ir asignando una etiqueta a cada muestra según características en común.

Seguidamente, se presentan los resultados de las métricas al entrenar un modelo con la representación latente del *autoencoder* normal. En la figura 18 se visualizan las métricas obtenidas para el conjunto de prueba.

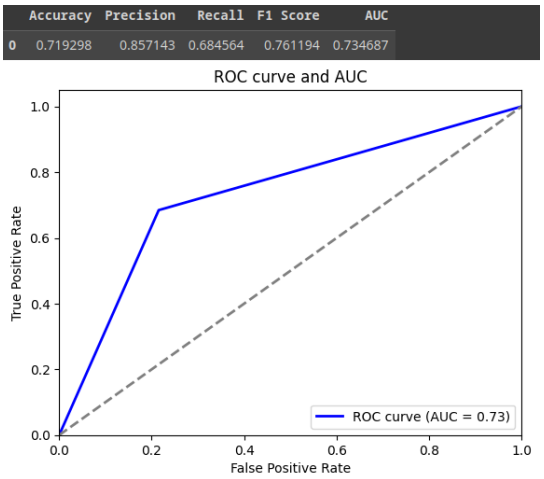

Figura 18. Resultados de métricas para el conjunto de prueba.

Utilizando el conjunto de entrenamiento se obtuvieron las métricas que se muestran en la figura 19.

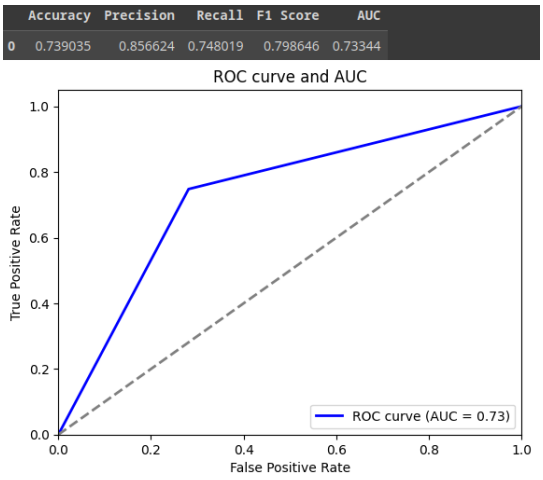

Figura 19. Resultados de métricas para el conjunto de entrenamiento.

A partir de las figuras anteriores, se puede observar que las predicciones obtenidas tienen métricas de precisión y AUC altas reflejando el correcto desempeño del modelo a la hora de clasificar si la planta tiene una enfermedad o no.

Al comparar los resultados para el conjunto de prueba y entrenamiento, se nota que no se presentó *overfitting* ni *underfitting* en las predicciones del modelo.

III-B2. Denoising Autoencoder: Seguidamente, se presentan los resultados relacionados con el *autoencoder* que tiene como entrada las imágenes con el filtro *salt and pepper*. En la figura 20, se visualiza el efecto de aplicar el filtro en algunas de las imágenes originales.

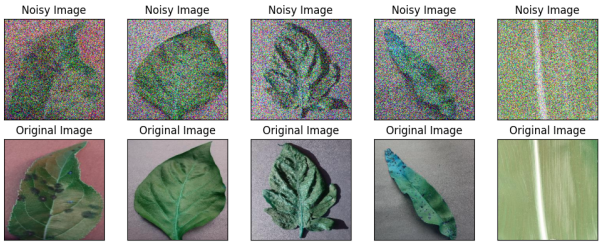

Figura 20. Resultado de la aplicación del ruido a las imágenes.

En la figura 21, se visualiza el resultado del PCA sobre el espacio latente. En este caso, se nota por las escalas que los clústers de puntos de cada color se forman de manera más cercana. Esto verifican que se están reconstruyendo de mejor manera las etiquetas, respecto al PCA del *autoencoder* normal.

Las figuras 22 y 23, se presentan los resultados para los conjuntos de *test* y *train* respectivamente.

Al analizar estos, se nota que, a pesar de que el PCA muestra clústers mejor formados, las métricas de precisión, *accuracy* y AUC reflejan que tanto para el conjunto de prueba como de entrenamiento, el modelo con ruido tuvo menor rendimiento que el *autodencoder* con las imágenes originales como entrada.

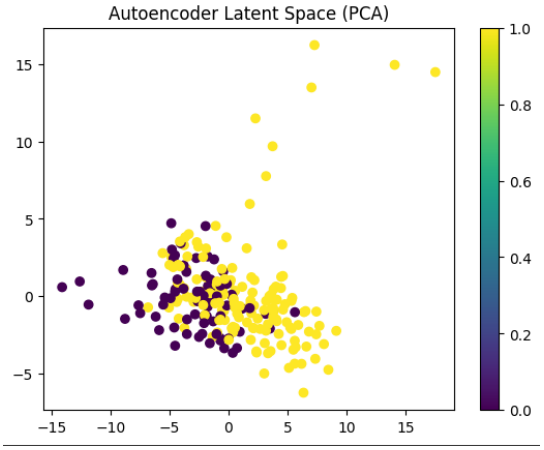

Figura 21. Representación del PCA para del *denoising* autoencoder.

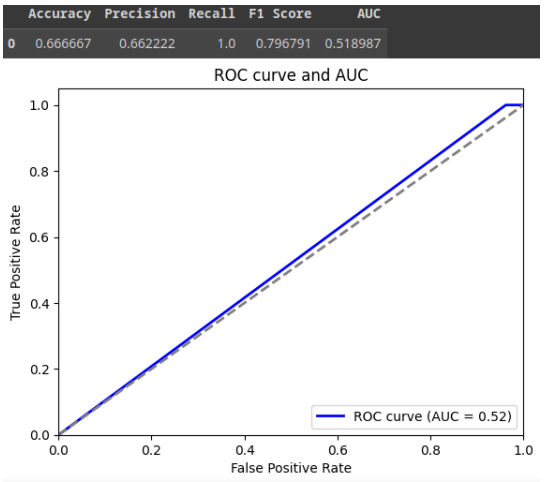

Figura 22. Resultados de métricas para el conjunto de prueba.

IV. Recomendaciones

En el caso de la implementación del *autoencoder* con y sin etiquetas, se puede entrenar un *autoencoder* para comprimir las imágenes de plantas y, posteriormente, utilizar las representaciones latentes (el código interno del *autoencoder*) como características para otras tareas.

En la misma línea, entrenar un *autoencoder* utilizando únicamente imágenes de plantas sanas y luego utilizar el modelo para reconstruir imágenes nuevas, puede indicar la presencia de una enfermedad o anomalía.

En el caso de tener etiquetas o clases en el conjunto de datos, los autoencoders con ruido *(denoising autoencoders)* se pueden utilizar para reconstruir imágenes dañadas o ruidosas, esto puede ayudar a mejorar la capacidad de generalización y robustez del modelo en la detección de enfermedades.

En el caso de este proyecto, el usar un red neuronal clasificadora, puede lograr beneficiar la capacidad de los *autoencoders* para extraer características relevantes y mejorar la precisión de la clasificación de enfermedades vegetales.

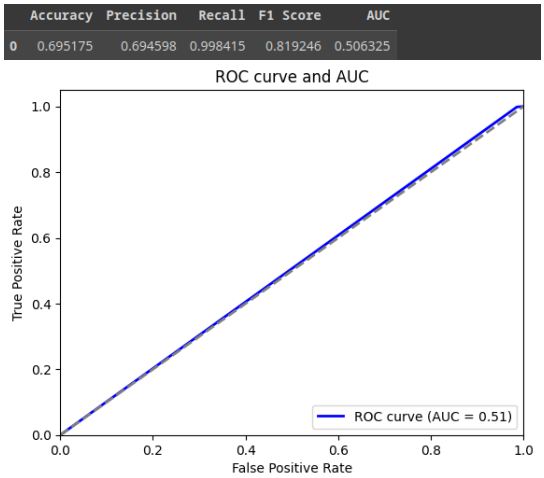

Figura 23. Resultados de métricas para el conjunto de entrenamiento.

V. Conclusiones

Al realizar el ciclo completo de diseño e implementación de las diferentes redes neuronales y según el análisis de los resultados en la sección III-A, se logró comprobar la validez de la hipótesis 1, ya que los modelos que utilizaron los espacios latentes como entrada en lugar de los datos originales, tuvieron un mejor desempeño general.

El mejor rendimiento se obtuvo con el modelo E, que estaba basado en el *autoencoder* con pesos congelados en la segunda ejecución.

Asimismo, al realizar el análisis presentado en la sección III-B, fue posible visualizar el comportamiento de los vectores latentes de los *autoencoders* al utilizar la técnica PCA de reducción de dimensionalidad.

Finalmente, respecto al análisis de la misma sección III-B, se concluye que no fue posible comprobar la validez de la hipótesis 2, ya que al agregar ruido a las imágenes del set de datos, no se mejoró el rendimiento de la clasificación del modelo.

Referencias

[1] Sharada P Mohanty, David P Hughes y Marcel Salathé. «Using deep learning for image-based plant disease detection». En: *Frontiers in plant science* 7 (2016), pág. 1419. DOI: 10.3389/fpls.2016.01419.

[2] TensorFlow. *Plant Village*. URL: https://www.tensorflow.org/datasets/catalog/plant_village.