Clasificación de imágenes de deforestación: redes neuronales convolucionales.

Juan Palmeros, David Damian, Brandon Hernandez

2022-12-13

Resumen

Este proyecto se desarrolló en el marco de la competencia *Schneider Electric Hackaton: Data Science Track*. En dicha competencia la tarea principal a resolver consistió en realizar una clasificación de distintos tipos de deforestación a partir de un conjunto de imágenes. Los resultados obtenidos para este proyecto no pudieron ser contrastados con los resultados esperados para la competencia, esto debido a que no se participó formalmente en la competencia ya que los horarios en los que se realizó difieren por mucho del uso horario en el que los integrantes del equipo se encuentran. Sin embargo, para generar la evaluación de los modelos y el cálculo de las métricas de desempeño, se utilizó un conjunto de etiquetas e imágenes que inicialmente pertenecían a nuestro conjunto de entrenamiento, por lo que se tuvo que reducir la cantidad de imágenes inicial para entrenar el modelo para poder crear un conjunto de imágenes de prueba.

Contexto teórico

La deforestación es la eliminación permanente de los bosques, que se produce por diversas razones y tiene muchas consecuencias negativas. La pérdida de árboles y otra vegetación puede provocar cambios climáticos, desertificación, erosión del suelo, menos cosechas, inundaciones, aumento de los gases de efecto invernadero en la atmósfera y otros problemas para los habitantes de las regiones afectadas. En los últimos 13 años, más de 43 millones de hectáreas de bosque han sido devastadas en el mundo (Schneider Electric Hackaton, 2022).

Una de las herramientas disponibles para combatir los problemas de deforestación es la identificación temprana de zonas deforestadas para evitar una expansión de este fenómeno a otras zonas aledañas. Así mismo, el uso de herramientas y algoritmos de inteligencia artificial, en conjunto con información pertinente que nos ayude a describir los distintos fenómenos de deforestación, pueden ayudar a la identificación temprana de las zonas deforestadas, por lo que es factible desarrollar algunas soluciones que faciliten dicha identificación y sean de fácil acceso para mitigar esta problemática.

Proceso generador de datos

Para este propósito, Schneider Electric puso a disposición un conjunto de imágenes proporcionadas por *Descartes Labs* que consistían en capturas satelitales de deforestación en el estado de Georgia. Las imágenes fueron distribuidas en un set de entrenamiento (1512 imágenes) y un set de prueba (635 imágenes), los cuales fueron creados de manera aleatoria. Así mismo, se proporcionó un archivo de texto que incluía las variables de latitud, longitud, y las etiquetas para cada una de las imágenes del conjunto de entrenamiento (Schneider Electric Hackaton, 2022).

Para este problema se eligió un enfoque de clasificación multiclase tomando en cuenta los distintos tipos de deforestación objetivo, los cuales se explican a continuación:

- Deforestación por plantaciones: se caracteriza por la presencia de plantaciones de árboles para actividades comerciales, tales como plantaciones de palma aceitera o árboles madereros que degradan la presencia de árboles locales.

Figure 1: Deforestación por plantaciones

- Deforestación por pastizales: Ocurre como consecuencia de la crianza de ganado, la cual describe áreas cubiertas únicamente por pastizales con pequeños arbustos o ausencia de árboles.

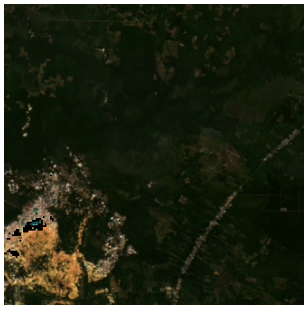

Figure 2: Deforestación por pastizales

- Deforestación por agricultura: ocurre en zonas en las que se realizan actividades de agricultura a gran escala.

Figure 3: Deforestación por pastizales

Análisis exploratorio de los datos

Para abordar correctamente el problema de clasificación, es necesario conocer la distribución de los datos que serán usados para la fase de entrenamiento de los modelos. En el caso de nuestro proyecto, existió un desbalanceamiento en la cantidad de imágenes por tipo de deforestación, donde puede observarse que la clase predominante es la de Plantación, mientras que la clase menos representada es la de pastizal.

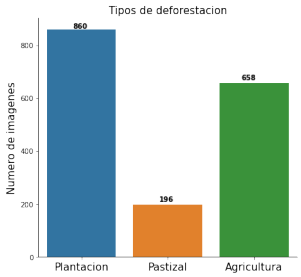

Figure 4: Distribucion de clases

Bajo esta problemática, se optó por aplicar técnicas de procesamiento de imágenes para generar nueva información a partir de la información original, con lo cual podemos tener una distribución de los datos más balanceada y agregar variabilidad en nuestro conjunto de datos. Sin embargo, una tarea previa a dicho procesamiento fue la de dividir el conjunto de imágenes disponible en un conjunto de entrenamiento y otro de prueba, ya que de esta manera se pueden evaluar los modelos y se pueden calcular las métricas de desempeño correspondientes. Partiendo de que se cuenta también con las respectivas etiquetas del tipo de deforestación para cada imagen, el conjunto de entrenamiento fue generado con el 80% de las imágenes con etiquetas originales, mientras que el conjunto de prueba fue generado con el 20% restante, ambos respetando la distribución original de los datos.

Posteriormente, se realizó el proceso de creación de imágenes para las clases que estaban menos representadas en comparación con la clase dominante, por lo que se optó por aplicar algunas transformaciones elementales (Tandia, 2021) al conjunto de imágenes de entrenamiento con miras a poder balancear la cantidad de imágenes entre clases. Las operaciones realizadas y su resultado sobre las imágenes originales se muestra a continuación:

1) Rotación: Giro aleatorio de la imagen entre 0 y 45 grados.

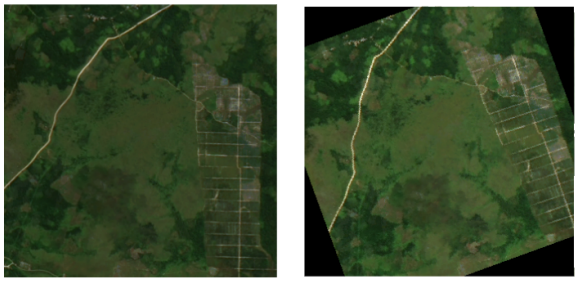

Figure 5: Transformacion de rotacion

2) Transposición vertical: Cambio en la orientación y orden de los pixeles de la imagen, de tal manera que los pixeles superiores de la imágen ahora se encuentren abajo, y viceversa.

Figure 6: Transformacion de transposicion vertical

3) Transposición horizontal: Cambio en la orientación y orden de los pixeles de la imagen, de tal manera que los pixeles derechos inviertan su posición con los pixeles izquierdos.

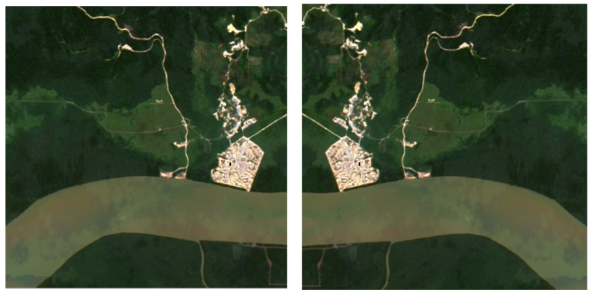

Figure 7: Transformacion de transposicion horizontal

Como resultado de las transformaciones anteriores, se logró obtener una distribución de las imágenes más balanceada en comparación con la distribución original que conformaba el conjunto de imágenes de entrenamiento.

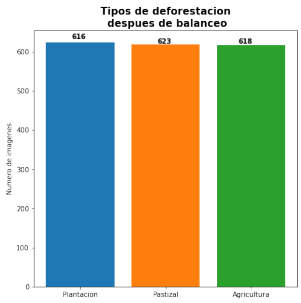

Figure 8: Distribucion de clases balanceadas

Aprendizaje supervisado y redes neuronales convolucionales

Dentro del aprendizaje de máquina existe un paradigma conocido como aprendizaje supervisado o aprendizaje con profesor, el cual se basa en la abstracción de un instructor que tiene el conocimiento o la información de interés, mientras que un aprendiz (algoritmo de aprendizaje) desconoce dicha información y tiene el propósito de aprenderla (Haykin, 1999). Para que el aprendiz pueda generalizar una solución, buscará ajustar sus parámetros característicos de acuerdo con la información proporcionada y una retroalimentación obtenida a partir de los aciertos y errores que cometa, la cual se conoce como fase de entrenamiento (Haykin, 1999). Como resultado de esta fase, se espera que el aprendiz pueda otorgar respuestas correctas ante información desconocida, o en su defecto identificar sus deficiencias para corregirlas en una siguiente etapa. La ventaja de este paradigma es la facilidad que otorga para optimizar los parámetros característicos del algoritmo de aprendizaje, llevándolo a cabo mediante una estrategia de retroalimentación o corrección del error (Haykin, 1999).

Concretamente, el aprendizaje supervisado permite modelar sistemas que establezcan relaciones de entrada y salida de información con base en ejemplos proporcionados para su aprendizaje. Durante la fase de entrenamiento, el algoritmo de aprendizaje es alimentado con parejas de datos de entrada y salida, permitiendo evaluar y corregir el funcionamiento del algoritmo, así como optimizar sus parámetros característicos de acuerdo con las respuestas obtenidas a la salida. Luego, durante la fase de pruebas, se usan los valores calculados para los parámetros característicos con la intención de obtener resultados ante nueva información desconocida, permitiendo evaluar el desempeño del algoritmo con dichos parámetros, y realizar mejoras en caso de ser necesario (Liu & Wu, 2012).

La siguiente imagen representa el paradigma del aprendizaje supervisado visto como un sistema de entrada-salida:

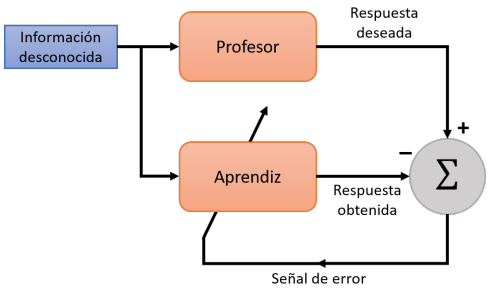

Figure 9: Paradigma de aprendizaje supervisado

Un algoritmo común en el aprendizaje supervisado son las Redes neuronales artificiales, las cuales pueden definirse como un algoritmo compuesto por unidades simples de procesamiento que, de forma paralela y distribuida, procesan cantidades masivas de información con la intención de almacenar conocimiento experimental y hacerlo disponible para las personas (Haykin, 1999). El modelo de una red neuronal artificial como un algoritmo de aprendizaje computacional inicia con la creación de las unidades simples de procesamiento, también llamadas perceptrones o neuronas. Los perceptrones cuentan con conexiones de entrada que permiten emular las sinapsis del cerebro, además de ser las encargadas de conectar las neuronas entre sí.

Los perceptrones también se componen por un conjunto de pesos de aprendizaje que están asociados a cada entrada de la neurona, los cuales son los parámetros característicos que ayudan a definir la fuerza con la que la neurona aprenderá la información recibida. Un elemento adicional para los perceptrones es el sumador, el cual tiene la tarea de realizar la suma aritmética de la multiplicación de cada valor de entrada recibido con su respectivo peso de aprendizaje. Por último, el perceptrón se compone también de una polarización o bias, la cual tiene la intención de incrementar o decrementar el acumulado del sumador y agregar variabilidad. Todos los elementos anteriores en conjunto permiten generar un resultado a la salida que será proporcionado a otra neurona para su aprendizaje.

La siguiente imagen representa la composición de un perceptrón con los elementos anteriormente descritos.

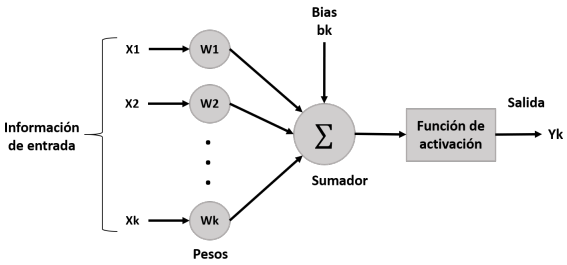

Figure 10: Composición de un perceptrón

Para que las redes neuronales aprendan y tengan el desempeño deseado, se requiere ajustar los pesos de entrada de cada neurona con base en los errores cometidos. Para ello, es necesario implementar un algoritmo de propagación hacia atrás que proporcione a las neuronas anteriores la información resultante a la salida de la red, lo que permite modificar los valores de dichos pesos de acuerdo con la comparación entre el valor esperado a la salida y el resultado obtenido (Haykin, 1999).

Un caso de aplicación de las redes neuronales son las Redes neuronales convolucionales, las cuales son un tipo de redes neuronales multicapa diseñadas para reconocer formas y patrones en información compuesta por dos o más dimensiones. La ventaja de usar este tipo de redes neuronales es la facilidad de poder trabajar con información caracterizada por altos niveles de distorsión, por ejemplo alta varianza o problemas de escalamiento o sesgo (Haykin, 1999). De acuerdo con Y. LeCun (1989), las redes neuronales convolucionales pueden definirse como simples redes neuronales que usan la operación de convolución en vez de realizar las operaciones aritméticas comunes en al menos una de sus capas.

Para llevar a cabo estas operaciones de convolución, se requiere de dos elementos principales: una imagen de entrada, y un kernel o filtro, de tal forma que el kernel sea el operador que modifique la imagen de entrada a la hora de realizar la convolución, permitiendo obtener características o patrones de las imágenes que puedan ser aprendidos por los perceptrones. LeCun y Bengio (1995) proponen tres operaciones elementales en las redes para el reconocimiento de patrones:

- Extracción de características: Cada neurona toma a su entrada la información generada por las neuronas de la capa anterior. Luego mediante la operación de convolución se extraen las características locales de la imagen de acuerdo con el kernel usado.
- Mapeo de características: Con la intención de reducir la cantidad de parámetros característicos a optimizar, las neuronas de la red son forzadas a transmitir los valores o pesos de cada kernel usado para cada mapa de características. Los mapas de características son una forma de representar bidimensionalmente la información que se va obteniendo como resultado de las operaciones de convolución, por lo que cada capa de la red está compuesta por múltiples mapas de características.
- Sub-muestreo o subsampling: Después de las operaciones efectuadas en una capa de convolución, se puede realizar un sub-muestreo de los mapas de características obtenidos, permitiendo reducir el tamaño de los mapas y disminuyendo la sensibilidad ante cambios o distorsiones

Un ejemplo de las operaciones anteriores aplicadas en una red neuronal puede observarse en la siguiente imagen, la cual busca reducir la complejidad de entendimiento de estos conceptos a través de una red de pocas capas.

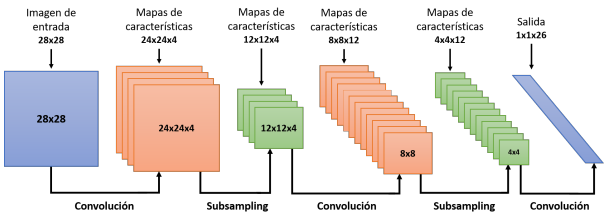

Figure 11: Operaciones de redes neuronales

Las operaciones aplicadas en la imagen anterior se enlistan a continuación:

1. Convolución a la imagen de entrada con 4 kernels de tamaño 5x5.
2. Subsampling para reducir de tamaño cada mapa de características a la mitad de su tamaño original.
3. Convolución a los mapas de salida del subsampling con 12 kernels de tamaño 5x5x4.
4. Subsampling para reducir el tamaño de cada mapa a la mitad de su tamaño de salida.
5. Convolución a los mapas de salida del subsampling anterior con 26 kernels de tamaño 4x4x12.
6. Unificación de los resultados obtenidos en un vector de probabilidades.

Transfer learning

Con la intención de reducir la complejidad del entrenamiento de las redes neuronales, una técnica comúnmente usada es la de transferencia de aprendizaje o *transfer learning*, la cual se basa en utilizar modelos previamente entrenados para un propósito específico y aprovechar todo el proceso de aprendizaje que tuvieron previamente. Este método resulta de gran utilidad para reducir el tiempo que pudiera tomar el entrenamiento de una red neuronal, así como usar menos datos y, por lo tanto, reducir la complejidad computacional que requeriría entrenar un modelo desde el inicio (Torrey & Shavlik, 2010).

La ventaja de usar esta técnica en nuestros desarrollos es aprovechar las características que adquirió previamente un modelo con un conjunto de imágenes distinto, facilitando la corrección de los valores de sus parámetros característicos y logrando un desempeño más fino para nuestros resultados. Particularmente en redes neuronales convolucionales permite reducir la complejidad que tomaría entrenar los modelos debido a la gran cantidad de imágenes que se pudieran requerir durante la fase de entrenamiento, al mismo tiempo que puede ser aplicado sin importar la arquitectura elegida o el conjunto de datos con el que se cuenta.

Modelos entrenados con ImageNet

Como se mencionó en la sección anterior, la intención de poder usar modelos pre-entrenados es aprovechar los parámetros característicos que fueron obtenidos durante una fase de entrenamiento previa, facilitando la generalización de nuestro problema y reduciendo el tiempo que pudiera tomar un entrenamiento desde el inicio. Para llevar a cabo esta tarea, se aprovecharon los modelos disponibles por el *framework Keras*, el cual es una capa de aplicación de TensorFlow que reduce la complejidad de entendimiento y creación de redes neuronales a través de instrucciones sencillas e intuitivas. Una de las ventajas de usar Keras es que cuenta con arquitecturas de modelos predefinidos y conjuntos de pesos adquiridos después de un proceso de entrenamiento con un conjunto de imágenes particular. Teniendo esto en cuenta, se aprovechó la capacidad de Keras para poder entrenar diversos modelos aprovechando los conjuntos de pesos generados después de entrenar cada modelo con el conjunto de imágenes ImageNet (Deng et. al., 2009), la cual es una base de datos con más de 14 millones de imágenes de objetos diversos que sirven para clasificar 1,000 tipos de objetos. Tomar como punto de partida los pesos generados para cualquier arquitectura entrenada con *ImageNet* resulta en un buen avance del proceso de extracción de características para cada neurona, ya que nos permite entrenar los modelos para para que únicamente ajustados al problema particular que se desea resolver y no deban realizar todo el cálculo completo de los parámetros característicos.

Modelos empleados para la tarea de clasificación

De las posibles arquitecturas a emplear disponibles en la *api* de Keras seleccionamos aquellas que ofrecían un buen balance de poder predictivo y costo computacional para el entrenamiento de la red. Se hicieron pruebas con 4 arquitecturas distintas: *ResNet50, DenseNet121, InceptionV3 y EfficentNetB3.*(Keras, 2022) En este texto reportamos y usamos únicamente los resultados de las dos arquitecturas con mejor desempeño en nuestro set de datos de validación: InceptionV3 y EfficientNetB3. Adicionalmente, se escogieron las dos arquitecturas mencionadas debido a que en la búsqueda que realizamos para encontrar referencias de problemas de clasificación encontramos que *InceptionV3* y *EfficientNetB3* eran arquitecturas que ya habían sido implementadas para un problema de clasificación de tipos de hojas y se logran desempeños aceptables (Kaggle, 2002).

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8

Figure 12: Tabla de arquitecturas de keras

Modelo InceptionV3

La arquitectura del modelo *InceptionV3* está conformada por 189 capas que realizan operaciones de: - Convolución, - Average Pooling: Reduce el tamaño de la imagen original a través de obtener el promedio de un área específica de la imagen, - MaxPooling: Reduce el tamaño de la imagen original a través de obtener el valor máximo de la intensidad de un área específica de la imagen input, - Concatenación: Transformación por medio de la cual se unen tensores de diferentes capas. - Dropout: Esta operación consiste en reducir aleatoriamente el número de nodos en la capa. Esta operación ayuda a controlar el sobreajuste del modelo. - Fully connected: Esta operación conecta todas las neuronas dentro de una capa con las neuronas de la siguiente capa.

Finalmente, en la última capa del modelo inception se obtiene un output denso mediante la activación softmax y en la cual se predicen 1,000 categorías de imágenes entrenadas con imagenet (Google Cloud, 2022).

El modelo InceptionV3 tiene un total 23,116,067 parámetros entrenables, para nuestro ejercicio de clasificación decidimos actualizar los pesos de los parámetros de las últimas 15 capas del modelo InceptionV3 lo cual resultó en un total de 1,708,163 parámetros entrenables. Como se muestra en la figura 13 el desempeño de este modelo en el set de validación no fue satisfactorio después de 50 épocas de entrenamiento. Se observa un sobreajuste en el modelo.

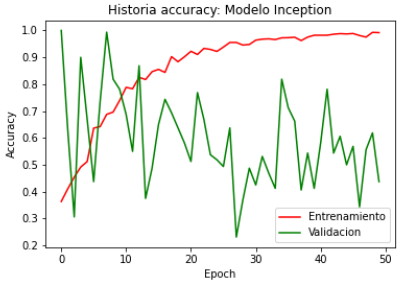

Figure 13: Historia Inception

Modelo EfficientNetB3

La arquitectura del modelo *EfficientNetB3* está conformada por 210 capas que realizan operaciones de: - Convolución, - Dropout, - Fully Connected, - Zero Padding: Operación que aplica filtros pero que devuelve un output de la misma dimensión que el input original por medio de la adición de ceros. - Normalización: Operación que normaliza el input en una capa determinada.

Finalmente, en la última capa del modelo inception se obtiene un output denso mediante la activación softmax y en la cual se predicen 1,000 categorías de imágenes entrenadas con imagenet. (Vardan, 2020)

El modelo EfficientNetB3 tiene un total 11,183,922 parámetros entrenables. Para nuestro ejercicio de clasificación decidimos actualizar los pesos de los parámetros de las últimas 15 capas del modelo EfficentNetB3 lo cual resultó en un total de 2,325,091 parámetros entrenables. Cabe mencionar que como medida de regularización empleamos el parámetro de dropout en la última capa entrenable de manera que aleatoriamente se desechan nodos.

Como se muestra en la imagen 15 el desempeño de este modelo en el set de validación fue considerablemente mejor que el modelo de EfficientNetB3, si bien aún se encuentra cierto grado de sobreajuste, las predicciones en el set de validación son mejores para este modelo, por lo cual lo escogimos como nuestra predicción final.

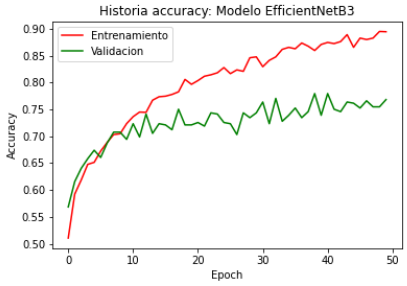

Figure 14: Historia EfficientNetB3

Resultados de los modelos

Resultados Inception

Una vez entrenado el modelo *Inception* actualizando los pesos de las 15 últimas capas obtuvimos resultados pocos satisfactorios principalmente para la clasificación de la primera categoría de deforestación. En particular, observamos que solo el 38% de los datos de validación fueron clasificados correctamente, lo cual está por debajo de un clasificador aleatorio para dicha categoría. De igual manera, el desempeño obtenido en la segunda categoría de deforestación fue de 51% de verdaderos positivos.

Metricas de desempeno: EfficientNetB3 Model				
		Precision	Recall	F1-score
	0	0.54	0.45	0.49
	1	0.53	0.4	0.46
	2	0.36	0.57	0.44
Accuracy	0.76			

Figure 15: Metricas de clasificacion

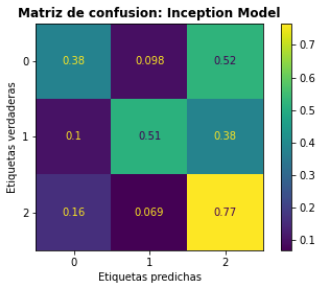

Figure 16: Matriz de confusion

Resultados EfficientNetB3

Al emplear la arquitectura de EfficientNetB3 con 15 capas de reentrenamiento obtuvimos mejores resultados en la clasificación de las categorías de deforestación. En particular, dado el *benchmark* establecido por parte de los organizadores de la competencia (*f1-score* igual o superior a 0.6 es considerado como un buen resultado) superamos dicho *benchmark* para cada una de las categorías individuales.

Metricas de desempeno: EfficientNetB3 Model				
		Precision	Recall	F1-score
	0	0.79	0.87	0.83
	1	0.84	0.7	0.76
	2	0.62	0.64	0.63
Accuracy	0.76			

Figure 17: Metricas de clasificacion

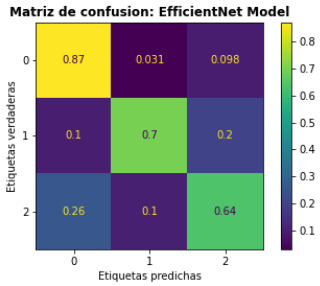

Figure 18: Matriz de confusion

Comparación de modelos

Como se comentó en la sección anterior, el modelo que tuvo mejores resultados fue el modelo de *EfficientNetB3*. Esto es cierto para todos los niveles de probabilidades, como se puede observar en las gráficas de la curva *ROC*. Así, para todos los niveles de probabilidades en los tres tipos de deforestación, logramos mejores niveles predictivos empleando EfficientNetB3.

Figure 19: Curvas ROC

Conclusiones

El desarrollo de este proyecto nos permitió lograr un entendimiento general sobre el funcionamiento de las redes neuronales convolucionales aplicadas a problemas de clasificación de imágenes. Un paso fundamental en este proyecto, así como en la mayor parte de los proyectos de ciencia de datos es la limpieza y entendimiento de los datos originales. En particular, para el proyecto lograr el balance entre las clases de nuestro problema fue esencial para lograr niveles de clasificación aceptables. Con respecto a las arquitecturas empleadas, si bien con el uso de la arquitectura logramos un desempeño aceptable, con la capacidad computacional suficiente podríamos reentrenar la arquitectura completa y actualizar todos los pesos de la red neuronal para obtener mejores métricas de clasificación.

Referencias

Agarwal, V. (Mayo 2020). Complete Architectural details of all EfficentNet Models. https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

Google Cloud (Consultado en Diciembre de 2022). Guia avanzada de InceptionV3. https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es-419#:~:text=Inception%20v3%20is%20an% 20image,multiple%20researchers%20over%20the%20years.

Haykin, S. (1999). Neural networks : a comprehensive foundation. Pearson Education. Kaggle (Abril 2022). Grapevine Image Dataset. https://www.kaggle.com/datasets/muratkokludataset/grapevine-leaves-image-dataset/code

Keras applications. (Consultado en Diciembre de 2022). https://keras.io/api/applications/ Lecun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.), Connectionism in perspective Elsevier.

Lecun, Y. & Bengio, Y. (1995). Convolutional Networks for Images, Speech, and Time-Series. The Handbook of Brain Theory and Neural Networks.

Liu, Q., & Wu, Y. (2012). Supervised Learning. Encyclopedia of the Sciences of Learning, 3243–3245. https://doi.org/10.1007/978-1-4419-1428-6_451

Schneider Electric Hackaton. (Noviembre 2022). Zero Deforestation Mission. https://nuwe.io/dev/challenges/data-science-se-european

Tandia, F. (Enero 2021). Some Tricks for Handling Imbalanced Dataset (Image Classification). https://www.linkedin.com/pulse/some-tricks-handling-imbalanced-dataset-image-m-farhan-tandia

Torrey, L., & Shavlik, J. (2010). Transfer Learning. Handbook of Research on Machine Learning: Applications and Trends, 242–264. https://doi.org/10.4018/978-1-60566-766-9.ch011