



**UNIVERSIDAD CENTRAL DEL ECUADOR**



**FACULTAD DE INGENIERÍA EN GEOLOGÍA, MINAS,  
PETRÓLEO Y AMBIENTAL  
CARRERA DE GEOLOGÍA**

**TEMA:**

**DETECCIÓN DE AFECTACIONES DERIVADAS  
DE LAS CORRIENTES COSTERAS MEDIANTE  
REDES NEURONALES CONVOLUCIONALES  
(CASO DE ESTUDIO “FENÓMENO DEL NIÑO”)**

**AUTORES:**

- Barragán Ramón
- Flores Alexander
- Galárraga Paulina
- Murillo Andrés

**ASIGNATURA:**

Software aplicado a la Geología.

**PROFESOR:**

Ing. Christian Mejía MSc.

**Quito – Ecuador**

## INDICE DE CONTENIDO

1.	INTRODUCCIÓN .....	4
2.	OBJETIVOS .....	4
2.1.	Objetivo General .....	4
2.2.	Objetivos Específicos .....	5
3.	TRABAJOS RELACIONADOS .....	5
4.	RECURSOS COMPUTACIONALES .....	6
4.1	Hardware .....	6
4.2	Software .....	7
	Lenguaje de Programación .....	7
	Librerías: .....	7
5.	METODOLOGÍA .....	9
5.1	Diagrama de Flujo .....	9
5.2	Dataset.....	10
5.3	Pre – Procesamiento: .....	12
5.4	División.....	12
5.5	Creación del Modelo .....	13
	Diagrama de la arquitectura.....	13
	Hiperparámetros del modelo .....	15
	Sumario de parámetros .....	16
5.6	Compilación y Ajuste .....	16
	Hiperparámetros de compilación .....	16
	Hiperparámetros de entrenamiento: .....	17
	Experimentos realizados y tiempos de ejecución: .....	17
	Gráficas de precisión y pérdida .....	18
5.7	Evaluación del Modelo .....	19
	Precisión y pérdida con datos de TEST .....	20
	Matriz de Confusión:.....	20
	Detección del cambio .....	21
5.8	Predicción.....	23
	Presentación de resultados.....	23
6.	CONCLUSIONES .....	25
7.	RECOMENDACIONES .....	26
8.	REFERENCIAS BIBLIOGRÁFICAS .....	27
9.	ANEXOS .....	29
	Librerías: .....	29
	Dataset: .....	29

Etiquetado: .....	31
División del Dataset en Training y Test: .....	32
Pre Procesamiento: .....	33

## ÍNDICE DE ILUSTRACIONES

Ilustración 1: Flujograma de trabajo .....	10
Ilustración 2: Búsqueda de imágenes satelitales mediante el WorldView .....	11
Ilustración 3: Distribución de carpetas en el Disco Local D.....	11
Ilustración 4: División de la Dataset en Jupyter .....	13
Ilustración 5: Creación del código del Modelo.....	13
Ilustración 6: Diagrama de la arquitectura de Red Convolutiva.....	14
Ilustración 7: Arquitectura de la Red Convolutiva.....	14
Ilustración 8: Representación total del modelo de red convolutiva.....	15
Ilustración 9: Sumatoria de parámetros .....	16
Ilustración 10: Modelo de Hiperparámetros de compilación .....	16
Ilustración 11: Declaración de Hiperparámetros de Entrenamiento.....	17
Ilustración 12: Ejecución de la red convolutiva con 8 épocas.....	18
Ilustración 13: Gráfica de precisión de entrenamiento y validación .....	18
Ilustración 14 Gráfica de pérdida de entrenamiento y validación.....	19
Ilustración 15: Evaluación del Modelo Creado .....	19
Ilustración 16: Matriz de confusión.....	20
Ilustración 17: Matriz de confusión. Fuente: <a href="https://rpubs.com/chzelada/275494">https://rpubs.com/chzelada/275494</a> .....	20
Ilustración 18: Modelo de Predicción.....	23
Ilustración 19: Presentación de resultados.....	24
Ilustración 20. Código de Importación de Librerías .....	29
Ilustración 21: Código del Dataset .....	30
Ilustración 22: Código del Etiquetado .....	31
Ilustración 23: Código de la División del Dataset en Training y Test .....	32
Ilustración 24: Código del Pre Procesamiento.....	33

## ÍNDICE DE TABLAS

Tabla 1: Estudios previos asociados a Machine Learning .....	6
Tabla 2: Características del equipo.....	7
Tabla 3: Descripción de librerías utilizadas en el proyecto .....	9
Tabla 4: Se presentan seis pruebas de validación del modelo al generar la imagen (blanco y negro) que corresponde a la detección de diferencias de entre la comparación de las imágenes pre_niño y post_niño.....	22

## **1. INTRODUCCIÓN**

Una de las influencias más directas en las variaciones climáticas anómalas y fortuitas es la elevación de temperatura de las corrientes marinas costeras. Estos fenómenos, al pasar por una porción de costa, derivan en la generación de lluvias anormalmente extensas, lo que puede producir inundaciones a mediano y largo plazo, lo cual conlleva a afectaciones focalizadas en locaciones costeras.

Estos eventos tienen una gran concurrencia en las costas ecuatorianas por lo que es necesario un sistema de monitoreo e identificación de las zonas donde ocurren y/u ocurrieron dichas afectaciones causadas por corrientes con temperatura elevada.

La zona a la cual se aplicará el caso de estudio para la determinación de afectaciones en el borde costero del Ecuador, será en las provincias de Manabí y Guayas. Estas provincias han sido, históricamente hablando, las más afectadas por el paso de una corriente cálida denominada el Fenómeno del Niño (E. Santana, 2011)

El propósito del presente trabajo es diseñar e implementar un programa que permita distinguir los cambios entre las fechas anteriores y posteriores al fenómeno del Niño en imágenes satelitales de la zona descrita aplicando Deep Learning, mediante Redes Neuronales Convolucionales, además de elaborar una base de datos extensa y confiable, para realizar la predicción de zonas afectadas a partir de nuevas imágenes satelitales.

Las ventajas de aplicar el Deep Learning es la confiabilidad que ofrecen los modelos de redes neuronales convolucionales para la detección de cambios, esto quiere decir, que los códigos de su arquitectura propuesta no tienen paso al error en los resultados, eso sí, cabe recalcar, que se debe realizar una investigación intensa del código que mejor cumpla los requerimientos y objetivos del proyecto propuesto. De esa manera, proveemos de un modelo de Deep Learning entendible y sobre todo útil para la detección de cambios causados por el fenómeno del niño, siendo más versátil esta técnica en contraste con la cartografía empleada como método tradicional.

## **2. OBJETIVOS**

### **2.1. Objetivo General**

Aplicar Redes Neuronales Convolucionales para la detección de afectaciones provocadas por el Fenómeno del Niño mediante imágenes satelitales.

## 2.2. Objetivos Específicos

- Elaborar una extensa dataset de imágenes satelitales de la zona costera del Ecuador propensa a ser afectada por el Fenómeno del Niño, obtenidas desde Google Earth, USGS/NASA, etc.
- Diseñar e implementar un modelo de Deep Learning para la detección de zonas de afectación por el Fenómeno del Niño utilizando redes neuronales convolucionales.
- Obtener zonas de afectación a partir de imágenes satelitales y contrastar con análisis convencionales.

## 3. TRABAJOS RELACIONADOS

La revisión de la literatura identificó que sí existen trabajos que se apoyan en el aprendizaje automático relacionados con la temática de afectaciones por fenómenos climáticos. Sin embargo, la gran mayoría de los artículos publicados tratan acerca del estudio de afectaciones provocadas por huracanes o sequías y un número reducido hablan sobre lluvias, siendo el contexto más cercano a nuestra información. Es así que podemos citar los siguientes:

Titulo	Autor	¿Que se hizo?	Diferencias con el presente trabajo
Re identificación de Personas Basada en Aprendizaje de Características de Partes del Cuerpo Mediante Redes Convolucionales en Triplet Loss	Durand, J (2018).	Aplicación de un modelo de redes convolucionales siamesas para comparar imágenes en tiempo real de personas, para un programa de reconocimiento facial, específicamente aplicando una técnica de Triple Loss.	La comparación mediante redes neuronales convolucionales, va a ser aplicada en imágenes satelitales. El reconocimiento será de los cambios asociados al antes y después del paso del fenómeno del niño.
Generación y Simulación de un modelo predictivo para prevenir inundaciones en viviendas aledañas a zonas de riesgo mediante técnicas de inteligencia artificial	Moreno, (2019)	Presenta la generación y simulación de un modelo predictivo para prevenir inundaciones en zonas de riesgo mediante técnicas de inteligencia artificial.	Nuestra investigación está enmarcada al uso de imágenes satelitales, que es un plus frente al manejo único de datos numéricos, para la generación de una tercera imagen que muestre los cambios detectados por el paso del Fenómeno del Niño
Optimization of neural network using kidney-		Aplicación de la optimización de redes neuronales	El presente proyecto está enfocado a la generación de una

inspired algorithm with control of filtration rate and chaotic map for real-world rainfall. Jaddi & Abdullah, (2017) convolucionales para el monitoreo de lluvias para la detección de cambios generados por el paso del Fenómeno del Niño en una imagen binaria. de un evento suscitado en el año 2017 que produjo una serie de lluvias anómalas.

Tabla 1: Estudios previos asociados a Machine Learning

En conclusión, estos presentes trabajos aportaron un gran conocimiento para dar un enfoque más centrado al entendimiento de redes neuronales convolucionales y, a su gran aplicación a la geología, en este caso a proponer zonas de cambios visibles por la afectación de fenómenos climatológicos. En la investigación de Moreno, (2019), permitió al grupo conocer la aplicación de las redes neuronales, los pasos secuenciales que se deben seguir para conseguir un modelo óptimo de arquitectura convolucional, mientras que, el trabajo de Jaddi & Abdullah, (2017) permitió entender como estos modelos de programación pueden ayudar para el monitoreo de lluvias y, de esta manera, asociarlo a nuestro proyecto.

#### 4. RECURSOS COMPUTACIONALES

El presente proyecto se lo realiza a través de ordenadores portátiles de gama media, el principal punto a evaluar en los ordenadores corresponde al almacenamiento, el procesador (Tabla 2). Todas estas características permiten que el ordenador se desarrolle de mejor manera en tanto a su rapidez de procesar y visualizar la información. Para una mejor comprensión y el aporte de ideas de manera grupal y en tiempo real, se trabajó con los códigos guardados en la nube de una forma segura.

##### 4.1 Hardware

Por la cantidad de imágenes que se requiere para conformar la base de datos y la calidad de resolución se ha tomado en cuenta como otro recurso un disco duro externo de 1 terabyte, con la finalidad de tener un respaldo de nuestra dataset.

Nombre del producto	Características
<b>Procesador</b>	Intel® Core™ i7-7500U CPU @2,70 GHz 2,90 GHz (7ma Generación)
<b>Tarjeta madre</b>	Pavilion 15-n 737984-501 737984-001.
<b>Memoria Instalada (RAM)</b>	8 GB de SDRAM DDR4-2133
<b>Disco duro</b>	SATA de 1 TB - 5400 rpm

<b>Disco Externo</b>	Toshiba Canvio Basics de 2,5" de 1TB
<b>Interfaz de red</b>	LAN 10/100/1000 GbE integrada

*Tabla 2: Características del equipo*

## 4.2 Software

El sistema operativo en el cual se ejecutaron los códigos es el Windows 7, que corresponde a una versión de Microsoft Windows de los sistemas operativos producidos por Microsoft Corporation. Este sistema operativo ha sido instalado en una PC tipo Hewlett-Packard (HP).

### *Lenguaje de Programación*

Para el desarrollo de clasificación multiclase se utilizó Notebook, es un entorno de trabajo interactivo que permite desarrollar código en Python de manera dinámica, a la vez que integrar en un mismo documento tanto bloques de código como texto, gráficas o imágenes.

El programa se ejecuta desde la aplicación web cliente que funciona en cualquier navegador estándar. El requisito previo es instalar y ejecutar en el sistema el servidor Jupyter Notebook. De esta manera se puede utilizar en un navegador o a través de suites, la más conocida es Anaconda, su distribución contiene lo necesario para una vez instalado Jupyter notebook se empiece a crear el código. De esta manera, el trabajo en Jupyter permitió importar las librerías necesarias para desarrollar el código deseado.

### *Librerías:*

<b>Nombre</b>	<b>Tipo</b>	<b>Descripción</b>
<b>Os</b>	Librería de Calculo numérico y análisis de datos en Python	Permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que nos refieren información sobre el entorno del mismo y permite manipular la estructura de directorios (para leer y escribir archivos).
<b>Re</b>	Librería de Python	Especifica un conjunto de cadenas que coinciden con ella; las funciones de este módulo permiten comprobar si una determinada cadena coincide con una expresión regular dada

<b>NumPy</b>	Librería de Calculo numérico y análisis de datos en Python	Proporciona una estructura de datos universal que posibilita el análisis de datos y el intercambio de datos entre distintos algoritmos. Además, proporciona funciones matemáticas de alto nivel que operan en estas estructuras de datos.
<b>Pandas</b>	Librería de Calculo numérico y análisis de datos en Python	Es una de las librerías de Python más útiles para los científicos de datos. Las estructuras de datos principales en pandas son Series para datos en una dimensión y DataFrame para datos en dos dimensiones.
<b>Matplotlib</b>	Librería de visualización de Python	Librería encargada de la generación de gráficos como: series temporales, histogramas, espectros de potencia, diagramas de barras etc., tanto en papel como digitalmente
<b>Seaborn</b>	Biblioteca construida sobre Matplotlib	Permite cambiar los estilos de una forma fácil, y agrega algunas visualizaciones de más alto nivel para gráficos comunes en estadística.
<b>Glob</b>	Herramienta incluida en la librería estándar de Python.	Proporciona una forma sencilla de acceder al contenido de un directorio desde un programa. Utiliza los caracteres comodines que suelen usarse en una consola de línea de comandos.
<b>Pathlib</b>	Python 3	Manipula rutas de sistemas de archivos de forma agnóstica en cualquier sistema operativo.
<b>PIL</b>	Recursos de Python	Edición de imágenes directamente desde Python. Soporta una variedad de formatos, incluidos los más utilizados como GIF, JPEG y PNG.
<b>Sklearn</b>	Recursos de Python	Fundamental escalar los datos para prescindir de las unidades de medida de las diferentes features.



<b>Keras</b>	Librería para Deep Learning de Python	Es un interfaz de alto nivel para trabajar con redes neuronales. Keras utiliza otras librerías de Deep learning (TensorFlow, CNTK o Theano) de forma transparente para hacer el trabajo que le digamos.
<b>Torch</b>	Interface de desarrollo Pytorch de Google Colab	La ventaja principal es el rango de probabilidades de salida. El rango será de 0 a 1, y la suma de todas las probabilidades será igual a uno.

*Tabla 3: Descripción de librerías utilizadas en el proyecto*

## 5. METODOLOGÍA

### 5.1 Diagrama de Flujo

La ilustración 1 muestra el diagrama de flujo que sintetiza los procedimientos a seguir en el proyecto. Empezando con la identificación y formulación del problema, para de esa manera direccionar la búsqueda y evaluación de información bibliográfica sobre la aplicación de redes neuronales en fenómenos climáticos, seguida por la obtención de imágenes satelitales, la separación de las mismas en pre\_niño y post\_niño para su respectivo procesamiento. A continuación, se realiza la búsqueda de librerías de Python amigables con “Colab” y Jupyter para la creación del modelo de red neuronal convolucional y, finalmente con la validación del modelo, se podrá obtener como resultado una tripleta de imágenes con la detección de cambios en forma binaria de una de ellas.

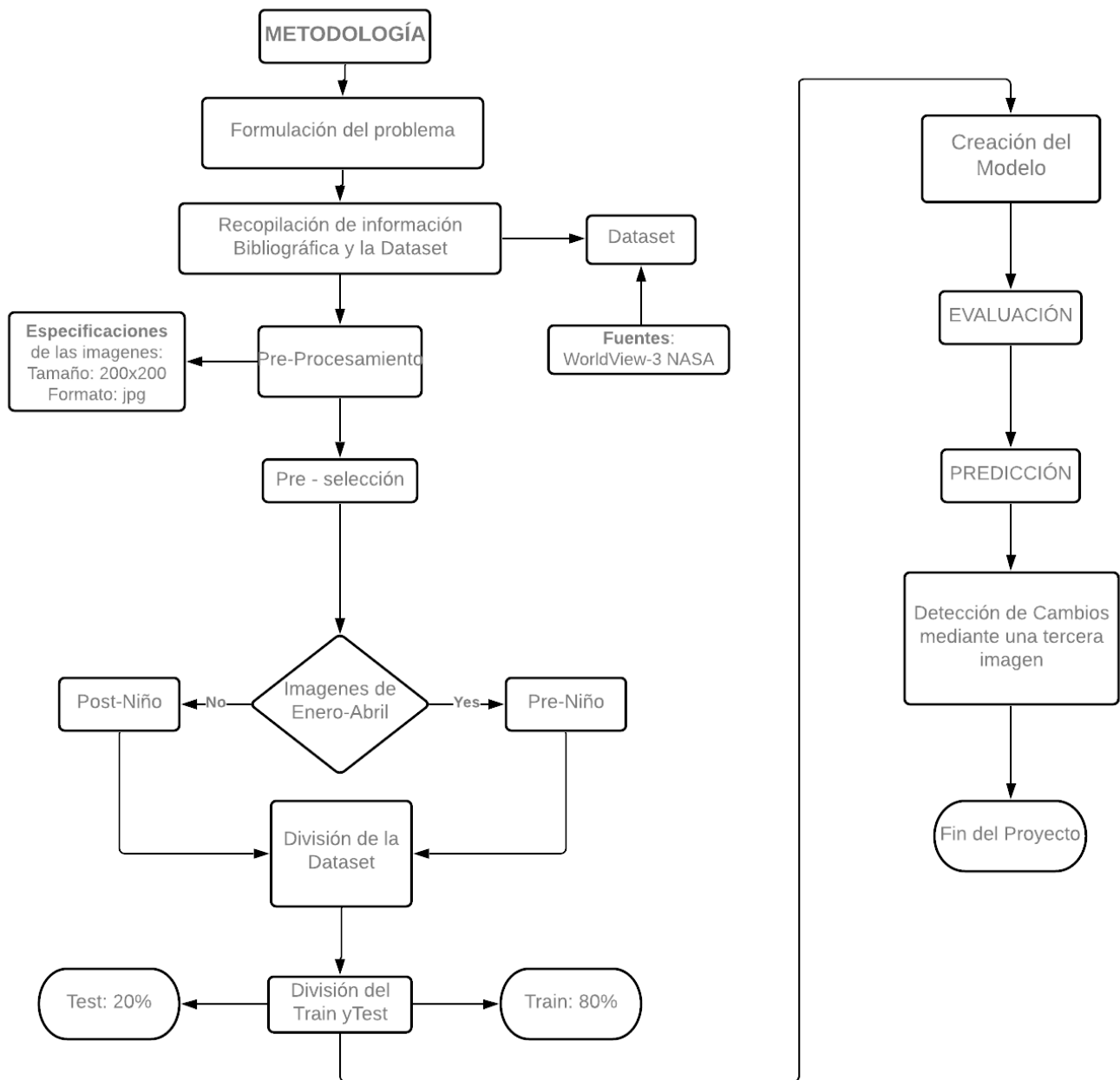
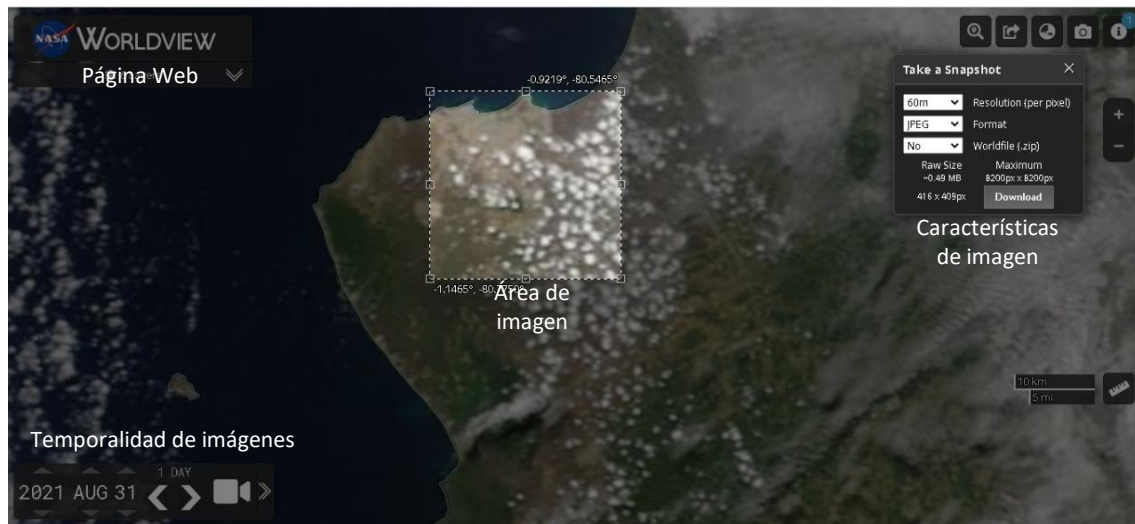


Ilustración 1: Flujograma de trabajo

## 5.2 Dataset

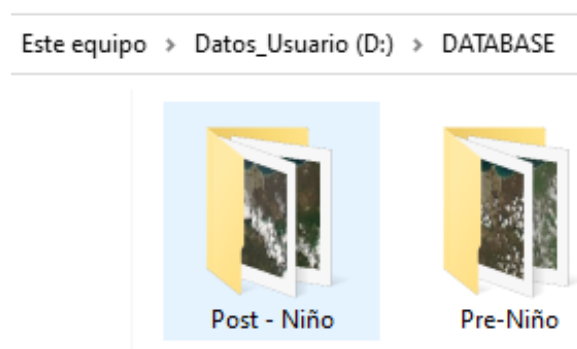
Las imágenes satelitales de la zona costanera de Manabí utilizadas en el modelo de clasificación para dos clases, se obtuvieron de una página proveedora de soluciones geoespaciales comerciales llamada Digital Globe, la cual mediante el último lanzamiento de su sexto satélite: WorldView-3 permite la observación de la Tierra y la descarga de imágenes satelitales de diferentes partes del mundo. Digital Globe en asociación con la NASA, concede la personalización de parámetros como la resolución y el formato de las imágenes, previo a su descarga.

Para la recopilación de la dataset, primero se procede a realizar un polígono de la zona de estudio que corresponde al área de la imagen que se desea obtener, en este caso, pertenece a la costa de Manabí y, finalmente, se establece el día, mes y año de cada imagen que se necesite descargar.



*Ilustración 2: Búsqueda de imágenes satelitales mediante el WorldView*

Las imágenes fueron descargadas en formato jpeg, como resultado de una búsqueda exhaustiva para cada mes, desde el año 2000 hasta julio del 2021, adquiriendo imágenes satelitales con un tamaño inicial de 416 x 409 píxeles, después mediante el uso de GIMP 2.10.24, se logró un tamaño final de 200 x 200 píxeles. En total se obtuvieron 861 imágenes satelitales divididas en 294 para los meses de enero, febrero, marzo y abril, conocido como el tiempo en el que no se da el paso del fenómeno del niño (pre\_niño) y 506 imágenes para los meses de mayo hasta diciembre, que representan este fenómeno (post\_ niño). Este grupo de imágenes fueron direccionadas al disco local D, en el cual se creó una carpeta con el nombre DATABASE, en esta carpeta se encuentran las dos categorías a clasificar: Pre\_Niño y Post\_niño.



*Ilustración 3: Distribución de carpetas en el Disco Local D*

### 5.3 Pre – Procesamiento:

La recopilación del dataset se construyó utilizando imágenes del Satélite WorldView-3. El WorldView-3 es el primer sensor de satélite comercial súper – espectral y de alta resolución que opere a una altitud estimada de 617 km, proveyendo de varias resoluciones tipo multiespectral y tamaños de imágenes de 300 x 500 píxeles. (Lunelli, 2014).

Este proyecto está enfocado en el uso de la arquitectura de una red convolucional, por lo que es necesario trabajar con imágenes de dimensiones entre 200 y 600 píxeles, (Atoany N. Fierro, 2019). Para lo cual, se obtuvieron imágenes que pasaron por un pre procesamiento manual en el programa GIMP 2.10.24 en donde se obtuvo un tamaño final de 200 x 200 píxeles y una resolución de 60m. Estas características permiten pronosticar de manera efectiva las diferencias causadas por el paso del Fenómeno del Niño como es nubosidad, precipitación, vegetación, inundaciones, etc. Entonces, se observa más densidad de características diferenciales en imágenes a gran escala, que, en imágenes a detalle, (Miguel Jiménez-Carrión, 2018) por lo cual se ha elegido un tamaño normalizado.

Para esta base de datos se ha utilizado 150 neuronas de salida en la red neuronal. En primera instancia se habían recolectado 861 imágenes para su tratamiento, pero debido a que se debe tener un número igual de imágenes en cada subcarpeta (pre\_niño y post\_niño), se trabajó con un total de 588, siendo divididas en 294 para cada subcarpeta, esto, con la finalidad de tener igual número de imágenes para el funcionamiento de la red neuronal y, las mismas deben ser suficientes para que el entrenamiento se pueda completar (Atoany N. Fierro, 2019).

### 5.4 División

La división de datos se realizó mediante la aplicación de la función **train\_test\_split()**, la cual permite dividir un conjunto de datos para dos propósitos diferentes, entrenamiento y prueba, **train\_test\_split()**, hará particiones aleatorias para los dos subconjuntos. Biblioteca tomada de: (N., 2018)

Las imágenes fueron divididas en un 80% (470 imágenes) y 20% (118 imágenes) para train y test, respectivamente. Este valor de 80% corresponde a que los efectos del niño no son iguales o estables en cada año (época) y, como la base de datos es bastante amplia temporalmente, entonces se presentarán varios efectos como: sequías, pluviosidad, nubosidad, vegetación, etc., debido a esto, es que la mayor cantidad de imágenes fueron destinadas para el entrenamiento y así evitar un “overffiting”, para que el programa reconozca solo como una lluvia o sequía el post niño.

Método de división: Train\_test\_split(), está conformado por siguientes parámetros.

- X, y. El primer parámetro es el conjunto de datos que está seleccionando usar.

- `train_size`. Este parámetro establece el tamaño del conjunto de datos de entrenamiento. Hay dos opciones: `Int` que requiere el número exacto de muestras, y `float`, que va de 0,1 a 1,0.
- `test_size`. Este parámetro especifica el tamaño del conjunto de datos de prueba. El estado predeterminado se adapta al tamaño del entrenamiento.

#### **DIVISIÓN DEL CONJUNTO DE DATOS EN ENTRENAMIENTO Y PRUEBA**

```
In [29]: train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)
print('Training data shape : ', train_X.shape, train_Y.shape)
print('Testing data shape : ', test_X.shape, test_Y.shape)

Training data shape : (470, 200, 200, 3) (470,)
Testing data shape : (118, 200, 200, 3) (118,)
```

*Ilustración 4: División de la Dataset en Jupyter*

### **5.5 Creación del Modelo**

En términos generales el modelo puede ser dividido en: entrada, proceso y salida. La entrada representada por la base de imágenes previamente etiquetadas y normalizadas, en proceso se encuentra las redes neuronales convolucionales, en el caso de este proyecto tres, que son las encargadas de identificar patrones o características, a mayor número de capas que este posea, mayor será la capacidad de la red en detectar patrones, como también, mayor será su precisión. Se aplicó un filtro de 32 para la red convolucional.

#### **CREACIÓN DEL MODELO**

```
In [35]: INIT_LR = 1e-3 # Valor inicial de learning rate. El valor 1e-3 corresponde con 0.001
#epochs = 10 # Cantidad de iteraciones completas al conjunto de imágenes de entrenamiento
#batch_size = 32 # cantidad de imágenes que se toman a la vez en memoria

In [70]: model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same', input_shape=(200, 200, 3)))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Dropout(0.5))
model.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(128, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(nClasses, activation='sigmoid'))

In [71]: model.summary()
```

*Ilustración 5: Creación del código del Modelo*

#### **Diagrama de la arquitectura**

En esta sección se plasma el diagrama de arquitectura del modelo establecido, para lo cual se necesita de las características que mejores resultados dieron y de funciones utilizadas, la mayoría de estas funciones son de la librería Keras. (Dhillon A, 2020)

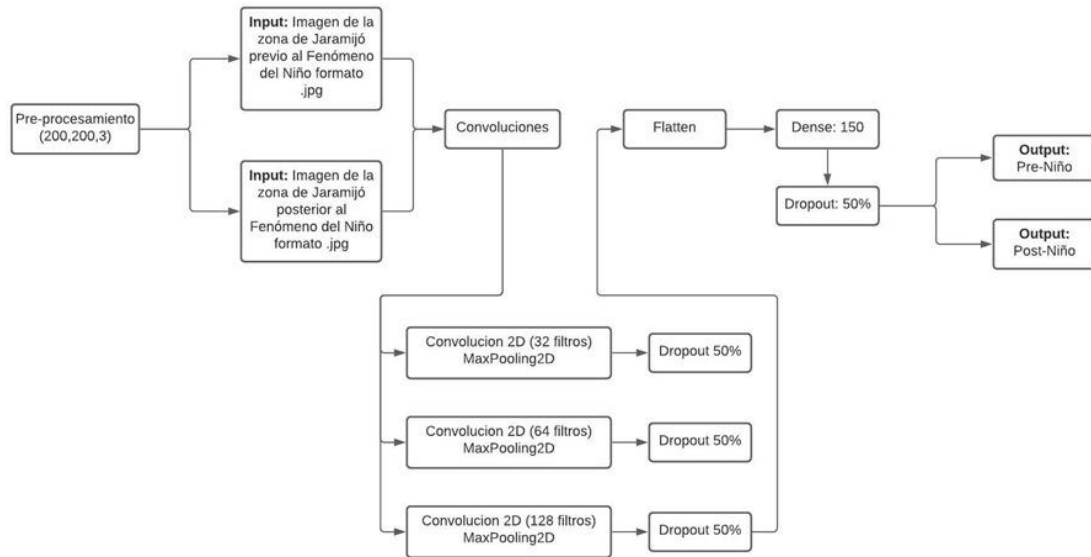


Ilustración 6: Diagrama de la arquitectura de Red Convolucional

En la ilustración 7 se muestran el número de parámetros que han dado como resultado en cada convolución realizada. Las operaciones realizadas de manera manual, se los detalla más adelante.

Model: "sequential\_8"

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 200, 200, 32)	896
max_pooling2d_15 (MaxPooling)	(None, 100, 100, 32)	0
dropout_15 (Dropout)	(None, 100, 100, 32)	0
conv2d_16 (Conv2D)	(None, 100, 100, 64)	18496
max_pooling2d_16 (MaxPooling)	(None, 50, 50, 64)	0
dropout_16 (Dropout)	(None, 50, 50, 64)	0
conv2d_17 (Conv2D)	(None, 50, 50, 128)	73856
max_pooling2d_17 (MaxPooling)	(None, 25, 25, 128)	0
flatten_8 (Flatten)	(None, 80000)	0
dense_17 (Dense)	(None, 150)	12000150
dropout_17 (Dropout)	(None, 150)	0
dense_18 (Dense)	(None, 2)	302
Total params: 12,093,700		
Trainable params: 12,093,700		

Ilustración 7: Arquitectura de la Red Convolucional

Se realizan tres convoluciones empezando con la primera: con un valor de fondo inicial correspondiente a las 3 bandas RGB, un Padding de 1 el cual establece el espacio de relleno requerido por todos lados de la imagen, esto evita la asignación de cada lado por separado y lo trata como un solo elemento, un valor de kernel de 3x3 y una red convolucional de 32 filtros sumado el valor del Bias (en función de los filtros) dando un resultado de 896 parámetros. Primero se realiza la función de activación de ReLu encargada de capturar un determinado o patrón de la imagen y llevarla a la siguiente red convolucional, seguido se aplica la función MaxPooling de 2x2 que aplica un filtro que reduce la dimensionalidad de la imagen, es decir, vuelve a las imágenes más pequeñas haciendo que la salida se reduzca a la mitad, en este caso se reduce a 100x100 pixeles, con el fin de reducir el coste computacional, tomando lo más relevante de cada píxel. Por último, se aplica el dropOut que se encarga de desactivar cierto porcentaje de neuronas durante el entrenamiento con el fin de evitar el sobre entrenamiento y, la aplicación del Flatting que aplanar a la matriz para un mejor análisis y aprendizaje del computador, donde cada característica se convierte en una entrada. Una vez aplicadas estas tres convoluciones se obtiene un valor final de 128 parámetros.

El presente modelo de red convolucional presenta dos imágenes de entrada, dos capas ocultas con tres neuronas y una imagen de salida que presenta la detección de cambios.

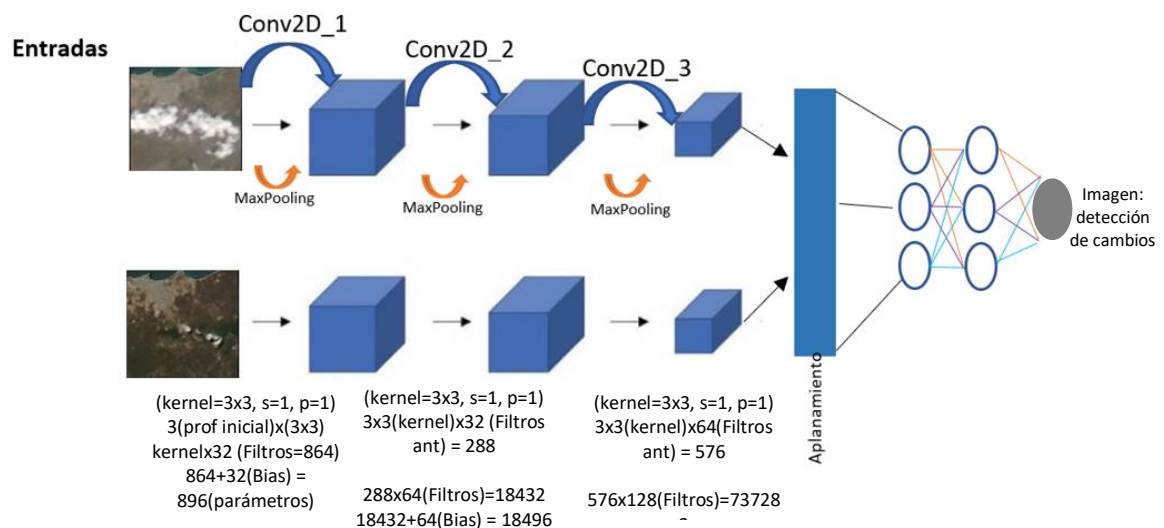


Ilustración 8: Representación total del modelo de red convolucional

### Hiperparámetros del modelo

En el modelo actual se usaron los siguientes Hiperparámetros:

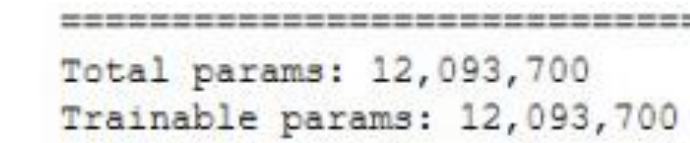


*Función de activación:* (Relu) se usa para evitar errores en el procesamiento de las imágenes al momento de que pueda haber valores negativos o erráticos que puedan influenciar en la interpolación y análisis del modelo (Programador\_Clic, 2021)

*Kernel:* se usa un tamaño de Kernel de 3x3 en el que se toman cuadrículas de 3x3 para tomar las imágenes con mayor detalle

*Padding:* same ya que se usa la misma imagen sin reducir píxeles en sus extremos, esto se realiza con el fin de mantener la imagen en detalle original

### *Sumario de parámetros*



```
=====
Total params: 12,093,700
Trainable params: 12,093,700
```

*Ilustración 9: Sumatoria de parámetros*

## **5.6 Compilación y Ajuste**

### *Hiperparámetros de compilación*

Los siguientes Hiperparámetros permitieron establecer el siguiente modelo de compilación de la ilustración 9.



```
In [72]: model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=['accuracy'])
```

*Ilustración 10: Modelo de Hiperparámetros de compilación*

*Función de pérdida – loss = “categorical\_crossentropy”*

Se utiliza como función de pérdida para el modelo de clasificación de clases múltiples donde hay dos o más etiquetas de salida. A la etiqueta de salida se le asigna un valor de codificación de categoría única en forma de 0 y 1. La etiqueta de salida, si está presente en forma de número entero, se convierte en codificación categórica usando keras.

Es una función de pérdida que se utiliza en tareas de clasificación de clases múltiples (Bueno, 2016). Estas son tareas en las que un ejemplo solo puede pertenecer a una de las muchas categorías posibles, y el modelo debe decidir cuál.

Formalmente, está diseñado para cuantificar la diferencia entre dos distribuciones de probabilidad.

*Optimizador = “adam”*



Adam es un algoritmo de optimización que se puede utilizar en lugar del procedimiento de descenso de gradiente estocástico clásico para actualizar los pesos de la red de forma iterativa en función de los datos de entrenamiento. (Torres, 2020)

Métrica = 'accuracy'

La precisión es una métrica para evaluar modelos de clasificación. De manera informal, accuracy es la fracción de predicciones que nuestro modelo acertó.

### ***Hiperparámetros de entrenamiento:***

Las siguientes funciones de la ilustración 11, permiten realizar el entrenamiento del modelo, mismos que son detallados a continuación.

```
batch_size = 64
epochs = 8
history = model.fit(train_X, train_label,
                    batch_size = batch_size,
                    epochs = epochs,
                    verbose=1,
                    validation_data = (valid_X, valid_label))
```

*Ilustración 11: Declaración de Hiperparámetros de Entrenamiento*

*batch\_size = 64*

Este es el número de ejemplos que se introducen en la red para que entrene de cada vez. Si el número es pequeño, significa que la red tiene en memoria poca cantidad de datos, y entrena más rápido (Artola, 2019). Sin embargo, es posible que no aprenda las características y detalles que pueden ser significativos en la predicción. Si es grande, ocurre, al contrario: es más probable que tenga en cuenta los casos más importantes a la hora de aprender, pero entrena más lento.

*epoch = 8*

Este es el número de veces que se van a pasar cada ejemplo de entrenamiento por la red. Es fácil caer en la tentación de pensar que un alto número de Epoch puede hacer que la red aprenda más fácil (Conte, 2018). En general, en determinado momento, el peso y el bias de cada neurona converge a un cierto valor para una Epoch concreta, y a partir de ahí además de consumir inútilmente tiempo y energía entrenándose, corre el riesgo de producirse overfitting. Por otro lado, un número bajo de Epoch puede provocar underfitting en la red, ya que no aprenda lo suficiente.

### ***Experimentos realizados y tiempos de ejecución:***

La ilustración 12 muestra el tiempo de ejecución de cada una de las épocas, donde el entrenamiento comenzó con un batch size = 64 y épocas = 8. A partir de la salida, podemos observar que la pérdida se acerca a un estado estable a medida que aumenta el número de épocas.

```

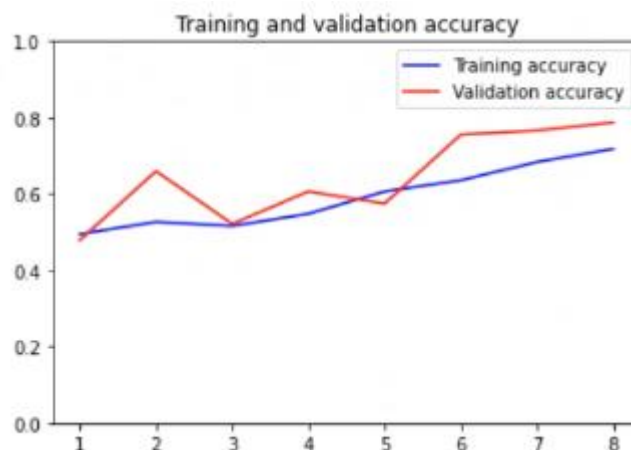
Epoch 1/8
6/6 [=====] - 24s 4s/step - loss: 5.0461 - accuracy: 0.5033 - val_loss: 0.6902 - val_
accuracy: 0.4681
Epoch 2/8
6/6 [=====] - 22s 4s/step - loss: 0.7521 - accuracy: 0.5265 - val_loss: 0.6933 - val_
accuracy: 0.4681
Epoch 3/8
6/6 [=====] - 22s 4s/step - loss: 0.6907 - accuracy: 0.5041 - val_loss: 0.6932 - val_
accuracy: 0.4681
Epoch 4/8
6/6 [=====] - 22s 4s/step - loss: 0.6902 - accuracy: 0.5219 - val_loss: 0.6931 - val_
accuracy: 0.4681
Epoch 5/8
6/6 [=====] - 21s 4s/step - loss: 0.6880 - accuracy: 0.5363 - val_loss: 0.6906 - val_
accuracy: 0.4681
Epoch 6/8
6/6 [=====] - 22s 4s/step - loss: 0.6878 - accuracy: 0.5195 - val_loss: 0.6892 - val_
accuracy: 0.4787
Epoch 7/8
6/6 [=====] - 22s 4s/step - loss: 0.6697 - accuracy: 0.5528 - val_loss: 0.6843 - val_
accuracy: 0.5957
Epoch 8/8
6/6 [=====] - 23s 4s/step - loss: 0.6534 - accuracy: 0.6191 - val_loss: 0.6453 - val_
accuracy: 0.7128

```

*Ilustración 12: Ejecución de la red convolucional con 8 épocas*

### **Gráficas de precisión y pérdida**

Las siguientes gráficas muestran en rendimiento comparable en sus curvas, tanto para el entrenamiento como en el conjunto de datos de validación, que es la razón por la cual presentan cercanía y similitud. Cada una presenta diferentes comportamientos conforme se ejecutan el número de épocas, para su debido entrenamiento.



*Ilustración 13: Gráfica de precisión de entrenamiento y validación*

En esta gráfica se puede observar un aumento paulatino en el porcentaje de precisión del entrenamiento, debido a que, conforme avancen los Epochs, el programa irá aprendiendo de mejor manera. En contraposición, esta gráfica presenta altibajos en ciertos puntos de las Epochs, esto se debe a que, los efectos del fenómeno del niño posteriores son variables dependiendo del año, en una época puede ocasionar sequía y en otra lluvia, es debido a esta ambigüedad de datos, que el porcentaje de precisión no es tan alto.

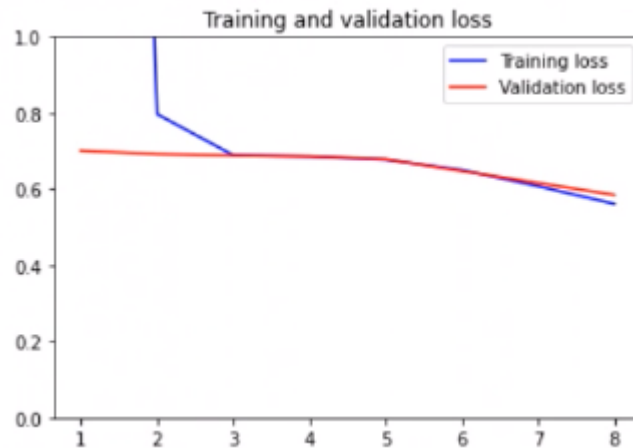


Ilustración 14 Grafica de pérdida de entrenamiento y validación

En esta grafica de perdida de la validación, inicialmente se mantiene constante, puesto que está analizando los datos, posteriormente, esta presenta una caída leve, lo cual indica disminución de errores. La grafica de pérdida del entrenamiento tiene un cambio más agresivo, puesto que al inicio de las Epochs tiene un 100 % de perdida, el cual disminuye inicialmente de manera abrupta para posteriormente presentar un decrecimiento suave.

## 5.7 Evaluación del Modelo

En la presente ilustración se muestra principalmente el modelo de evaluación de la prueba con un tiempo de duración de 2s y 409ms, presentando las funciones loss y accuracy.

```

EVALUACIÓN DEL MODELO

In [21]: test_eval = model.evaluate(test_X, test_Y_one_hot, verbose=1)
4/4 [=====] - 2s 409ms/step - loss: 0.6161 - accuracy: 0.7458

In [22]: print('Test loss:', test_eval[0])
          print('Test accuracy:', test_eval[1])

Test loss: 0.6160521507263184
Test accuracy: 0.7457627058029175

```

Ilustración 15: Evaluación del Modelo Creado

En la evaluación del modelo se presenta un porcentaje de 61% de pérdidas, esto debido a la ambigüedad de los datos en cuanto al fenómeno del niño, lo cual puede resultar confuso para el programa. Adicionalmente, se tiene un porcentaje de precisión del 75%, que, si bien es alto debido a la similitud, en características de las imágenes, no puede alcanzar valores más altos, debido a las razones previamente descritas.

### ***Precisión y pérdida con datos de TEST***

En la ilustración 16 se observa la precisión, el Recall, score y support de las clases obtenidas para el proyecto, cada una se definirá a continuación:

```
In [83]: target_names = ["Class {}".format(i) for i in range(nClasses)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

	precision	recall	f1-score	support
Class 0	0.85	0.64	0.73	64
Class 1	0.67	0.87	0.76	54
accuracy			0.75	118
macro avg	0.76	0.76	0.75	118
weighted avg	0.77	0.75	0.74	118

*Ilustración 16: Matriz de confusión*

Para identificar las clases de pre\_niño y post\_niño se muestran los valores de las métricas: Accuracy de 75%, esto quiere decir que, de las predicciones que se realizaron, el 75% de interacciones arrojaron una clase correcta; la métrica de Precisión refleja mayor porcentaje en la clase 0 (85%) que corresponde a Pre\_niño, es decir, esta clase presenta mayor cantidad de interacciones correctas, mientras que, la clase 1 correspondiente a Post\_niño, presenta una precisión menor con el 67%. Recall muestra un valor de 87% en la clase 1, por lo tanto, las imágenes de la clase 1 fueron identificadas y asignadas correctamente a su tipología (predicción correcta de valores verdaderos).

### ***Matriz de Confusión:***

Evalúa el rendimiento de un modelo de clasificación, comparando los valores reales con los predichos por el modelo, considerando como ejemplo la siguiente ilustración, se tiene que las columnas representan valores reales y las filas valores predichos, las variables de destinos son valores positivos o negativos.

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

*Ilustración 17: Matriz de confusión. Fuente: <https://rpubs.com/chzelada/275494>*

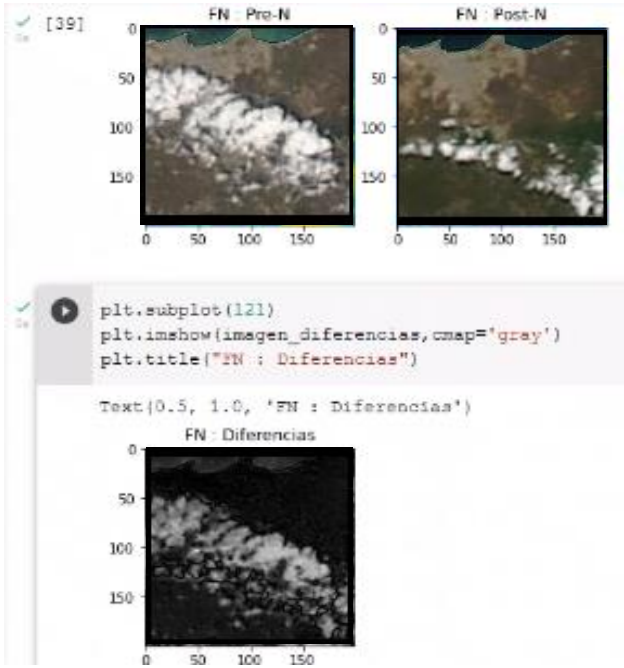
Las letras TP representa un valor verdadero positivo (El valor real fue positivo y el modelo predijo un valor positivo), FP (Falso positivo), el valor real fue negativo pero el modelo predijo un valor positivo), FN (Falso negativo), el valor real fue positivo, pero el modelo predijo un valor negativo

y finalmente TN (Verdadero negativo), el valor real fue negativo y el modelo predijo un valor negativo (Barrero, 2015).

*Detección del cambio*

A continuación, se realizaron seis pruebas de funcionamiento del modelo de red convolucional:

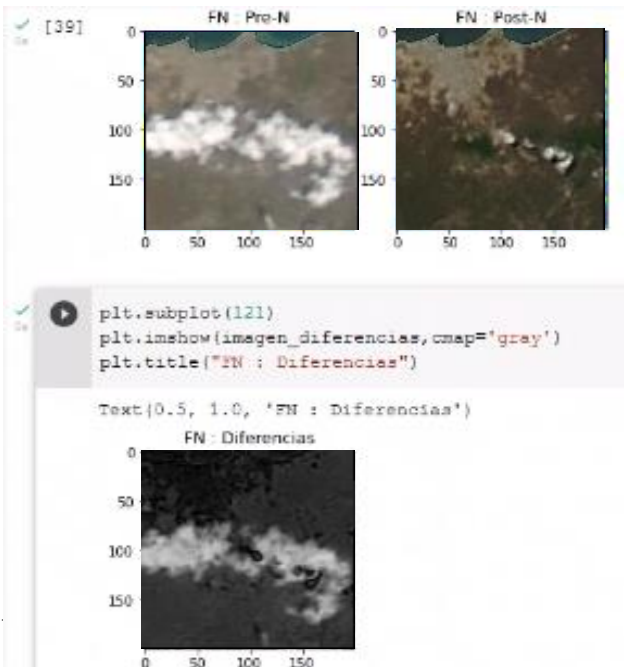
**PRUEBA 1**



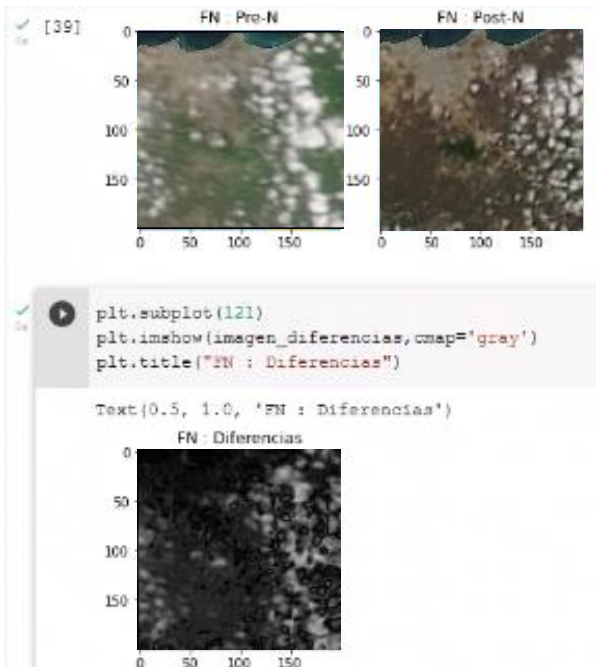
**PRUEBA 2**



**PRUEBA 3**



**PRUEBA 4**



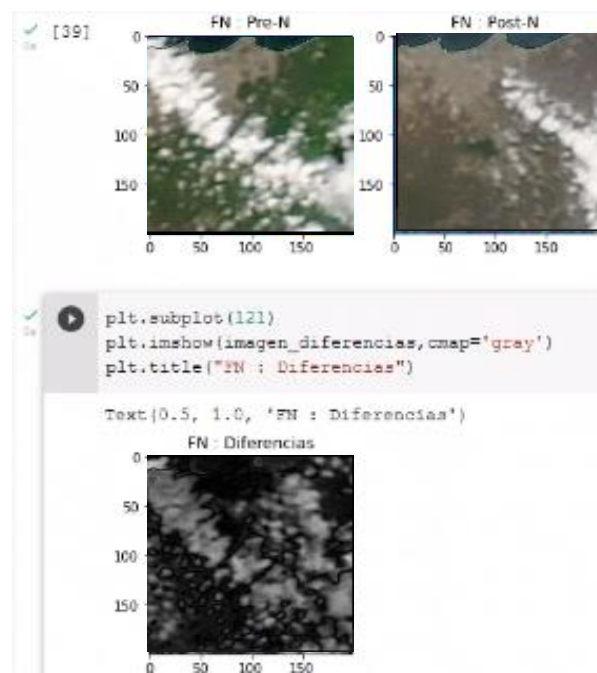
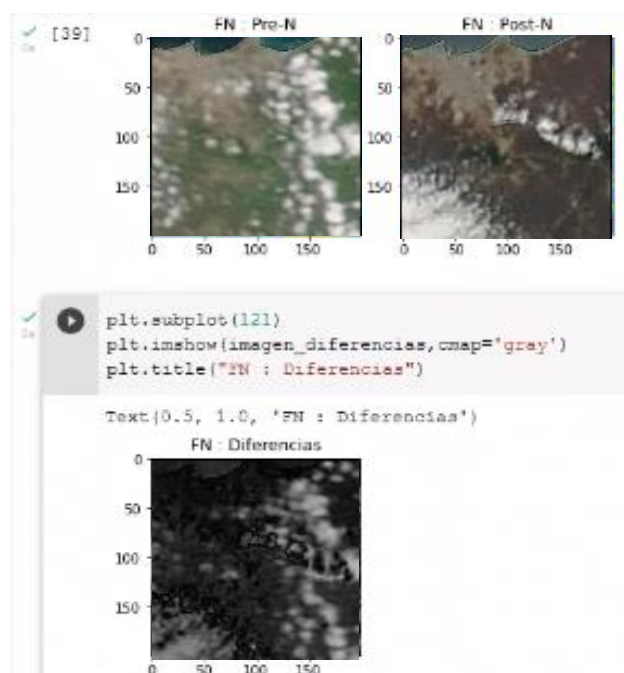


Tabla 4: Se presentan seis pruebas de validación del modelo al generar la imagen (blanco y negro) que corresponde a la detección de diferencias de entre la comparación de las imágenes pre\_niño y post\_niño.

En la presente tabla se puede observar cuatro pruebas gráficas de la validación del modelo de convolución, para la detección de cambios por el paso del Fenómeno del Niño. Se ha generado una imagen en la escala de grises para cada prueba, la cual representa las diferencias detectadas entre una comparación de dos imágenes (Pre\_Niño y Post\_Niño) mediante la creación del modelo convolucional. Las zonas en blanco (escala muy clara de grises) de cada imagen representan cambios de tipo morfológico y climático (nubosidades, sequías, inundaciones), este último, causado principalmente por altas precipitaciones o períodos de sequía. Y, en negro, corresponde a las zonas en las que el modelo no ha detectado cambios.

Se presencia una ambigüedad (de forma, más no de fondo) en el resultado de la imagen detectora de cambios, ya que dependiendo del año y del período (meses) el paso del fenómeno del niño puede generar períodos de fuertes lluvias o en su defecto desecamiento. También hay que considerar si el período en el que se presenta el Fenómeno del Niño corresponde a una época humedad, semi húmeda o árida, dependiendo de aquello, este cambio climático llegará a ser muy versátil en cuanto a sus afectaciones. Sin embargo, el propósito del modelo convolucional es la detección de cambios representado en una imagen de salida.



## 5.8 Predicción

A continuación, se muestra el modelo del código para la predicción:

```
from skimage.transform import resize

images=[]
# AQUI ESPECIFICAMOS UNAS IMAGENES
filenames = ['C:\\PROYECTO SOFTWARE\\Predicción\\pred10.jpg']

for filepath in filenames:
    image = plt.imread(filepath,0)
    image_resized = resize(image, (200, 200),anti_aliasing=True,clip=False,preserve_range=True)
    images.append(image_resized)

X = np.array(images, dtype=np.uint8) #convierto de lista a numpy
test_X = X.astype('float32')
test_X = test_X / 255.
plt.subplot(121)
plt.imshow(test_X[0,:,:], cmap='gray')
predicted_classes = model.predict(test_X)

for i, img_tagged in enumerate(predicted_classes):
    print(filenames[i], FN[img_tagged.tolist().index(max(img_tagged))])
```

*Ilustración 18: Modelo de Predicción*

Finalmente, una vez entrenado el modelo, se procede a realizar la prueba del mismo, en la cual se cargaron dos imágenes indistintamente. El modelo recibió las imágenes y definió que correspondía a los efectos de sequía Post-Niño posiblemente provenientes de los años 2008 o 2010.

### ***Presentación de resultados***

Los resultados que se observa en esta ilustración 19, es producto de un proceso de creación, procesamiento, evaluación y predicción de un modelo de red neuronal tipo convolucional, con el fin de extraer las características más relevantes de un par de imágenes que representan diferentes periodos de tiempo (en meses y años), en un producto de salida que representa la detección de dichas características definidas como los cambios que se ha producido en una misma zona por la influencia del Fenómeno del Niño.

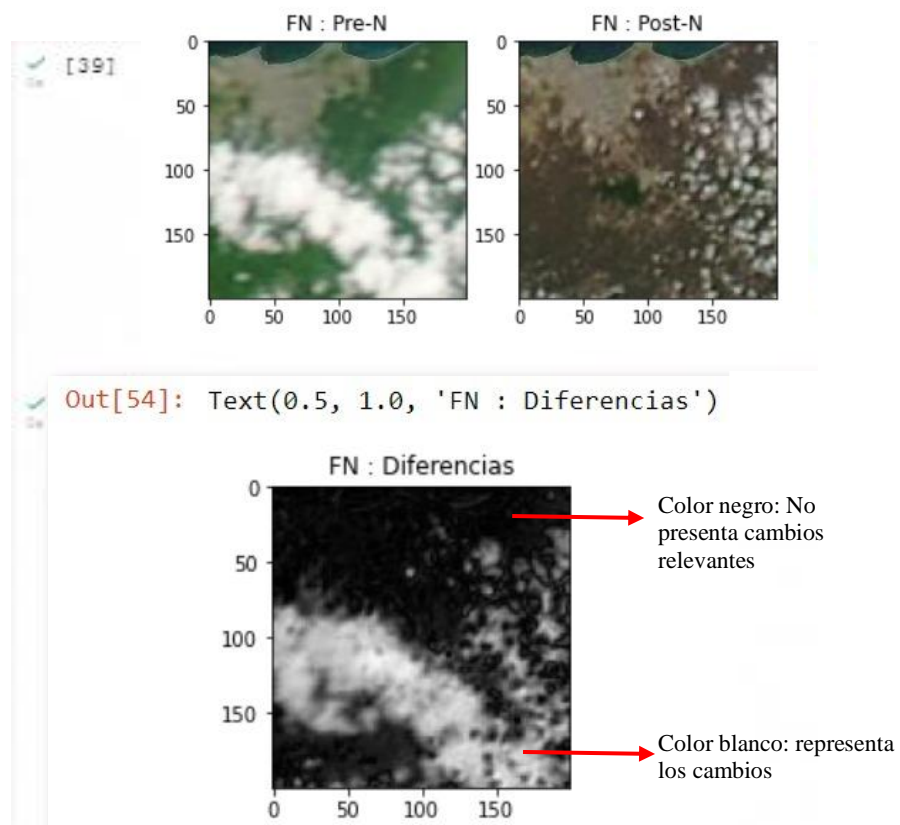


Ilustración 19: Presentación de resultados



## 6. CONCLUSIONES

La aplicación de Deep Learning en la influencia de las corrientes climáticas costeras, permite generar un modelo para la detección de cambios derivados a la influencia del fenómeno del niño, el cual extrae las características más importantes de una extensa dataset, para de esta manera, generar una imagen en blanco (cambios) y negro (sin cambios). La implementación de este modelo, es innovar la forma de estudio de las afectaciones causadas por cambios climáticos bruscos, para poder identificar las zonas vulnerables y en conjunto con la gestión de riesgos, disminuir el desastre causado por dichas afectaciones.

El entrenamiento de un modelo es de vital importancia y mientras más veces se lo haga será mejor y se obtendrán mejores resultados, de lo contrario, el modelo caerá en el overffiting, y únicamente será capaz de reconocer datos de entrada particulares que le hayamos enseñado y no estará preparado para manejar nuevos datos.

Se logró una DATASET confiable pero que no es muy extensa, debido a la complejidad en cuanto a la detección de características y el modelo de red neuronal propuesto. Sin embargo, las primeras 861 imágenes recolectadas nos fue de mucha ayuda para realizar las pruebas necesarias y concluir que se debía disminuir esta cantidad en 588 imágenes totales, con el fin de mejorar el proceso de ejecución del modelo y, evitar imágenes duplicadas que puedan sobre entrenar nuestro modelo.

Se diseñó un modelo Deep Learning con un código previamente realizado, pero adaptado hacia los objetivos de este proyecto, desde 0 con características propias, de tal manera que los parámetros del modelo final fueron: 3 convoluciones, 128 filtros, 150 neuronas, 8 épocas y división del DATASET en (80 - 20%). Estos parámetros fueron obtenidos tras un proceso de prueba y error de varios ensayos.

Los parámetros aplicados al proyecto son de vital importancia porque de estos depende la predicción de las nuevas imágenes, Si está establecido correctamente, se reflejará en gráficas de pérdida y precisión.

## **7. RECOMENDACIONES**

Se recomienda obtener más imágenes en un año en específico, debido a que los efectos del fenómeno del niño varían por cada temporalidad (meses y años), es decir, no son necesariamente cíclicos. Debido a esto los datos se vuelven ambiguos y tienden a variar en sus predicciones.

Para futuros trabajos relacionados a esta investigación, se recomienda establecer un modelo adicional que permita identificar el tipo de afectación, en donde se presente una imagen de salida con los colores correspondientes a cada cambio, que pueden ser entre los principales: inundaciones, nubosidades, sequías, hundimientos.

Es recomendable aplicar un dataset extenso y realizar un entrenamiento con más épocas para que la predicción tenga mayor precisión.

## 8. REFERENCIAS BIBLIOGRÁFICAS

- Artola. (2019). *Clasificación de imágenes usando redes neuronales convolucionales*. Sevilla. Obtenido de <http://bibing.us.es/proyectos/abreproy/92402/fichero/TFG-2402-ARTOLA.pdf>
- Atoany N. Fierro, M. N. (2019). *Redes Convolucionales Siamesas y Tripletas para la Recuperación de Imágenes Similares en Contenido*.
- Barrero, G. (2015). *Modelos de Machine Learning para analisis de Calidad de Software RPG*. Lima-Perú.
- Bueno, V. M. (2016). *Estudio de técnicas de aprendizaje automático basado en redes neuronales para reconocimiento biométrico de personas*.
- Conte, D. (2018). *Artificial con TensorFlow para predicción de*. Sevilla. Obtenido de <http://bibing.us.es/proyectos/abreproy/91796/fichero/TFG-1796-CONDE.pdf>
- Dhillon A, V. G. (2020). *Red neuronal convolucional: una revisión de modelos, metodologías y aplicaciones para la detección de objetos*. Obtenido de <https://link.springer.com/article/10.1007%2Fs13748-019-00203-0>
- E. Santana, J. D. (2011). *LOS EFECTOS DEL FENOMENO EL NIÑO EN LA OCURRENCIA DE UNA ALTA TASA DE EROSION COSTERA EN EL SECTOR DE PUNTA GORDA, ESMERALDAS*.
- Joao Felipe Pimentel, L. M. (2016). *A Large-scale Study about Quality and Reproducibility of Jupyter Notebooks*. New York.
- Lunelli, P. (19 de Agosto de 2014). *MundoGEO*. Obtenido de <https://mundogeo.com/es/2014/08/19/digitalglobe-lanza-con-exito-el-satelite-worldview-3-de-alta-resolucion/>
- M.H. Glantz, R. K. (1997). *UNA VISIÓN GENERAL DEL FENÓMENO EL NIÑO, OSCILACIÓN SUR (ENOS)*.
- Martínez, A. H. (2019). *Estudio e implementación de un sistema de detección de puntos significativos en objetos estructurados con CNNs*. Obtenido de [https://ebuah.uah.es/dspace/bitstream/handle/10017/39953/TFM\\_Hernandez\\_%20Martinez\\_2019.pdf?sequence=1&isAllowed=y](https://ebuah.uah.es/dspace/bitstream/handle/10017/39953/TFM_Hernandez_%20Martinez_2019.pdf?sequence=1&isAllowed=y)
- Miguel Jiménez-Carrión, F. G.-S. (2018). *Modelado y Predicción del Fenómeno El Niño en Piura, Perú mediante Redes Neuronales Artificiales usando Matla*. Piura.

- N., S. (2018). *Un análisis de redes neuronales convolucionales para la clasificación de imágenes*. *International Conference on Computational Intelligence and Data Science*. ELSEVIER. Obtenido de <https://www.sciencedirect.com/science/article/pii/S1877050918309335>
- Programador\_Clic*. (2021). Recuperado el 17 de Septiembre de 2021, de <https://programmerclick.com/article/255571158/>
- señales, D. d. (2019). Recuperado el 17 de Septiembre de 2021, de [https://www.ecorfan.org/proceedings/Proceedings\\_Ciencias\\_de\\_la\\_Tierra\\_Fisica\\_y\\_Matematicas\\_TI/Proceedings\\_Ciencias\\_de\\_la\\_Tierra\\_Fisica\\_y\\_Matematicas\\_TI\\_4.pdf](https://www.ecorfan.org/proceedings/Proceedings_Ciencias_de_la_Tierra_Fisica_y_Matematicas_TI/Proceedings_Ciencias_de_la_Tierra_Fisica_y_Matematicas_TI_4.pdf)
- Torres, J. (2020). *Python Deep Learning. Introducción práctica con Keras y Tensorflow 2*. MARCOMBO, S. L. España. Obtenido de [https://torres.ai/wp-content/uploads/2020/02/Empieza\\_a\\_leer\\_Python\\_Deep\\_Learning.pdf](https://torres.ai/wp-content/uploads/2020/02/Empieza_a_leer_Python_Deep_Learning.pdf)

## 9. ANEXOS

### *Librerías:*

```
import numpy as np
import os
import re
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import sklearn as skplt

import keras
#from keras.utils import to_categorical
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, Input, Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
#from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
```

*Ilustración 20. Código de Importación de Librerías*

### *Dataset:*

```
dirname = os.path.join(os.getcwd(), 'C:\PROYECTO SOFTWARE\Dataset_EN')
imgpath = dirname + os.sep

images = []
directories = []
dircount = []
prevRoot=''
cant=0

print("leyendo imagenes de ",imgpath)
```

```

for root, dirnames, filenames in os.walk(imgpath):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant=cant+1
            filepath = os.path.join(root, filename)
            image = plt.imread(filepath)
            images.append(image)
            b = "Leyendo..." + str(cant)
            print (b, end="\r")
            if prevRoot !=root:
                print(root, cant)
                prevRoot=root
                directories.append(root)
                dircount.append(cant)
                cant=0
dircount.append(cant)

dircount = dircount[1:]
dircount[0]=dircount[0]+1
print('Directorios leidos:',len(directories))
print("Imagenes en cada directorio", dircount)
print('Suma Total de imagenes en subdirectorios:',sum(dircount))

leyendo imagenes de C:\PROYECTO SOFTWARE\Dataset_EN\
C:\PROYECTO SOFTWARE\Dataset_EN\Post-N 1
C:\PROYECTO SOFTWARE\Dataset_EN\Pre-N 294
Directorios leidos: 2
Imagenes en cada directorio [295, 293]
Suma Total de imagenes en subdirectorios: 588

```

*Ilustración 21: Código del Dataset*

### ***Etiquetado:***

```
labels=[]
indice=0
for cantidad in dircount:
    for i in range(cantidad):
        labels.append(indice)
        indice=indice+1
print("Cantidad de etiquetas creadas: ",len(labels))
```

Cantidad de etiquetas creadas: 588

```
FN=[]
indice=0
for directorio in directories:
    name = directorio.split(os.sep)
    print(indice , name[len(name)-1])
    FN.append(name[len(name)-1])
    indice=indice+1
```

0 Post-N

1 Pre-N

```
y = np.array(labels)
X = np.array(images, dtype=np.uint8) #convierto de lista a numpy

# Find the unique numbers from the labels
classes = np.unique(y)
nClasses = len(classes)
print('Total number of outputs : ', nClasses)
print('Output classes : ', classes)
```

Total number of outputs : 2

Output classes : [0 1]

*Ilustración 22: Código del Etiquetado*



### *División del Dataset en Training y Test:*

```
train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)
print('Training data shape : ', train_X.shape, train_Y.shape)
print('Testing data shape : ', test_X.shape, test_Y.shape)
```

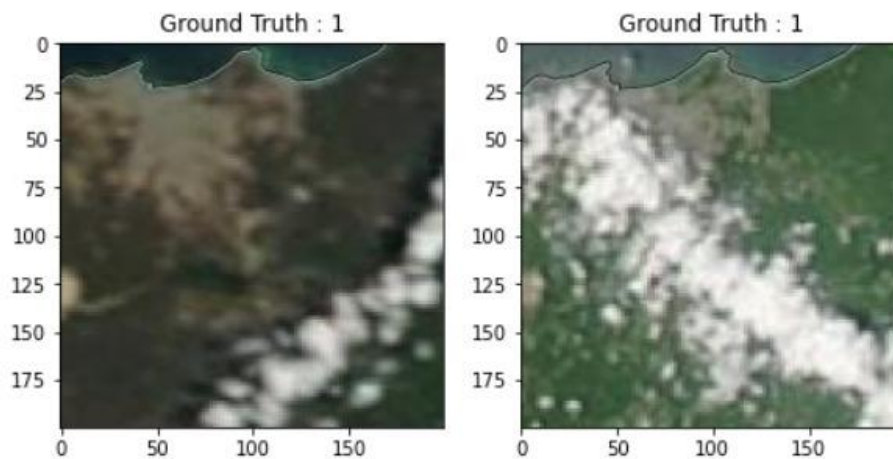
Training data shape : (470, 200, 200, 3) (470,)  
Testing data shape : (118, 200, 200, 3) (118,)

```
plt.figure(figsize=[8,8])

# Display the first image in training data
plt.subplot(121)
plt.imshow(train_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(train_Y[0]))

# Display the first image in testing data
plt.subplot(122)
plt.imshow(test_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(test_Y[0]))
```

Text(0.5, 1.0, 'Ground Truth : 1')



*Ilustración 23: Código de la División del Dataset en Training y Test*



### *Pre Procesamiento:*

```
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.
```

```
# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Display the change for category label using one-hot encoding
print('Original label:', train_Y[0])
print('After conversion to one-hot:', train_Y_one_hot[0])
```

```
Original label: 1
After conversion to one-hot: [0. 1.]
```

```
#Mezclar todo y crear los grupos
train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.2, random_state=42)

print(train_X.shape,valid_X.shape,train_label.shape,valid_label.shape)

(376, 200, 200, 3) (94, 200, 200, 3) (376, 2) (94, 2)
```

*Ilustración 24: Código del Pre Procesamiento*