UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA EN GEOLOGÍA, MINAS, PETRÓLEOS Y AMBIENTAL

ESCUELA DE GEOLOGÍA

SOFTWARE APLICADO A LA INGENIERÍA

Aplicación de redes neuronales convolucionales siamesas para la detección de cambios por el proceso de erosión regresiva del Río Coca

Docente: Ing. Christian Mejía.

GRUPO N° 1

Autores:

- ✓ Cabezas Paúl
- ✓ Falconí Anthony
- ✓ Flores Katherine
- ✓ Pilamunga Karen

Curso:

- ✓ Noveno Semestre

Quito – Ecuador

ÍNDICE

1. INTRODUCCIÓN

Los procesos de erosión regresiva son eventos particulares de erosión hídrica controlada principalmente por la naturaleza del lecho rocoso y la capacidad erosiva de un sistema fluvial, de manera que materiales susceptibles a ser erosionados facilitan la infiltración de agua (Stocking & Murnaghan, 2003), en consecuencia, la remoción paulatina del material en superficie. El Río Coca presenta actualmente un evento de erosión regresiva que se ha evidenciado a partir de la pérdida de la cascada San Rafael, este fenómeno natural presenta un problema debido a la amenaza que representa para las poblaciones e infraestructuras cercanas al área de influencia (Figura 1). La Inteligencia Artificial permite automatizar la información imágenes y sonidos además de datos para ser procesados mediante un ordenador, con el fin de solucionar problemas de manera óptima y en menor tiempo. (Alcaide, 2020). Las redes neuronales convolucionales siamesas permiten identificar los cambios entre un par de imágenes que tengan la misma geometría y resolución. Es así que la aplicación de estas redes ayudaría a realizar un análisis multitemporal del proceso de erosión regresiva del Río Coca a partir del uso de ortofotografías. El proceso tradicional contempla mayor tiempo en el análisis de cambios a partir de pares de fotografías, y se fundamenta en el manejo de las bandas de las imágenes y aplicación de filtros para reconocer variaciones en las imágenes, mientras que la aplicación de un modelo de aprendizaje requiere un menor tiempo en obtener los mismos resultados.

Figura 1.Mapa de ubicación del área de estudio.

2. OBJETIVOS

2.1 Objetivo General

Aplicar la Inteligencia Artificial para la evaluación del proceso de erosión regresiva del Río Coca a través del tiempo mediante el uso de redes neuronales convolucionales siamesas.

2.2 Objetivos Específicos

Recopilar imágenes multitemporales del Río Coca a través de la plataforma del Geoportal IGM que permitan la elaboración de una base de datos adecuada para ser procesada en una red neuronal convolucional.

Diseñar e implementar una red neuronal convolucional siamesa capaz de identificar los cambios en la morfología de un mismo escenario geológico pero distintos cronológicamente.

Demostrar la funcionalidad del modelo mediante la predicción de imágenes distintas a las usadas en el aprendizaje.

3. TRABAJOS RELACIONADOS

La información en cuanto a proyectos relacionados con la detección de cambios es muy escasa, y solo existe un amplio interés en la aplicación de redes neuronales siamesas para el análisis biométrico destinado al reconocimiento facial y establecer un grado de similitud entre dos personas, para definir si se trata de la misma.

Los proyectos de aprendizaje profundo son muy complejos, existen repositorios de parpes los cuales son muy pocos en los que se aplican las redes neuronales siamesas y más aún códigos de modelos existentes en la web.

Tabla 1. Trabajos Relacionados.

Nombre del estudio	**Autor**	**¿Qué hicieron?**	**Diferencia con nuestro trabajo**
Redes convolucionales siamesas y tripletas para la recuperación de imágenes similares en contenido	Fierro et al., (2019).	Utilizó redes múltiples siamesas y tripletas para cumplir con el objetivo; la arquitectura siamesa tiene dos redes Alexnet idénticas que se componen por 5 capas convolucionales y 3 capas superiores totalmente conectadas, además se aplica una función de costo contrastiva. El entrenamiento de las capas convolucionales se basa en un modelo pre-entrenado ImageNet y se reentrenan las capas superiores utilizando la base de datos objetivo	No se utilizará tripletas y solo se construirá una red siamesa por la cual ingresan un par de imágenes.
Clasificación y mapeo automático de coberturas del suelo en imágenes satelitales utilizando Redes Neuronales Convolucionales	Suárez et al., (2017).	En este trabajo se presenta un método de aprendizaje automático basado en redes neuronales convolucionales de arquitectura tipo ConvNet para la clasificación automática de coberturas del suelo a partir de imágenes Landsat 5 TM. La ConvNet fue entrenada a partir de las anotaciones manuales por medio de interpretación visual sobre las imágenes satelitales con las que los expertos generaron el mapa de cobertura del parque nacional el Tuparro, de los Parques Nacionales Naturales de Colombia.	No se usará imágenes Landsat, se utilizará ortofotografías, además no se realizará interpretaciones visuales, se imprimirá un mapa de cambios.
Redes Neuronales Convolucionales Siamesas aplicadas a la Verificación Facial	Alcaide et al., (2020).	Se realiza un análisis sobre el diseño e implementación de las técnicas más potentes en biometría facial. Se propone y explica una nueva técnica utilizada para el diseño de las redes de neuronas convolucionales siamesas, muy eficiente para este tipo de problemas, utilizando el estado del arte de distintas arquitecturas que se analizan, y la implementación de cada una de ellas.	En nuestro proyecto se quiere tener como resultado una tercera imagen que detecte las diferencias entre dos imágenes.

Redes Neuronales Siamesas	Mingote V et al., (2016)	Este trabajo plantea la elaboración de un sistema de reconocimiento biométrico de personas basado en redes neuronales profundas que utiliza como característica biométrica una imagen digital del rostro humano con la que se pueda realizar la identificación facial de dicha persona.	Se utilizará fotografías con el fin de estudiar y caracterizar procesos geológicos, más no para reconocer características biométricas.

La información bibliográfica revisada es muy limitada, debido a que su enfoque de estudio se centra en proyectos de identificación de rostros para los cuales se dispone de una base de datos muy extensa, además la objetividad de estos proyectos es el determinar si estas imágenes mantienen similitud, más no generan una imagen que refleje cambios.

4. RECURSOS COMPUTACIONALES

4.1 Hadware

Tabla 2. Recursos computacionales de Hardware.

Nombre del dispositivo	LENOVO
Tipo	Laptop
CPU	
Procesador	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
Núcleos	Dual Core
Procesador de gráficos	NVIDIA GeForce GeForce GTX 1660
	2 GB dedicated
Unidad de Almacenamiento	
RAM	16 GB
Disco duro	Disco duro SSD de 237 GB Disco duro HDD de 931 GB
Comunicaciones	
Tarjeta de red	Ethernet
Internet	Netlife de 40 MB

4.2 Software

- Sistema Operativo: Windows 10.
- Lenguaje de Programación: Python.
- Microsoft Edge o navegadores

Microsoft Edge es un navegador web libre y de código abierto desarrollado para distintas plataformas

- Anaconda-Jupyter Notebook

  Jupyter Notebook proporciona un entorno que permite a los usuarios escribir, documentar y ejecutar código, visualizar datos, realizar cálculos y ver los resultados. Concretamente, la fase de prototipado incluye la ventaja de que el código se organiza en celdas independientes, es decir, es posible probar bloques concretos de código de forma individual. Gracias a que existen muchos kernels o núcleos adicionales, Jupyter no se limita al lenguaje de programación Python, lo que aporta muchísima flexibilidad a la hora de crear código y de hacer análisis.

4.3 Librerías

- *Matplotlib.pyplot:* Librería especializada en la creación de gráficos en dos dimensiones. (Sánchez, 2020). Permite crear y personalizar los tipos de gráficos más comunes.
- *PIL:* Python Imaging Library es una librería gratuita para el lenguaje de programación Python que agrega soporte para abrir, manipular y guardar muchos formatos de archivos de imágenes diferentes. Sirve para procesar imágenes y que se lean a manera de información (Salcedo, 2017).
- *Numpy:* Es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación (Sánchez, 2020).
- *Os:* El módulo os de Python le permite a usted realizar operaciones dependientes del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc. Este módulo tiene

métodos para ver variables de entornos del Sistema Operativo con las cuales Python está trabajando en muchos más (Caballero, 2020).

- *Tensorflow:* Es una biblioteca de software de código abierto para computación numérica, que utiliza gráficos de flujo de datos. Los nodos en las gráficas representan operaciones matemáticas, mientras que los bordes de las gráficas representan las matrices de datos multidimensionales (tensores) comunicadas entre ellos. Además, es una gran plataforma para construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos (Buhigas, 2018).
- *Keras:* Es una biblioteca de redes neuronales artificiales de código abierto. Está desarrollada en Python y puede ejecutarse sobre diferentes plataformas como TensorFlow o Theano. Keras está diseñado para ir construyendo por bloques la arquitectura de cada red neuronal, incluyendo redes convolucionales y recurrentes, que son las que permiten, junto a los bloques "más tradicionales", entrenar modelos deep learning (Piperlab, 2020).
- *Imageio:* Es una librería para python que proporciona una interfaz para leer y escribir imágenes (Buthi, 2020).

5. METODOLOGÍA

La metodología utilizada durante todas las etapas del proyecto se describe en la figura 2.

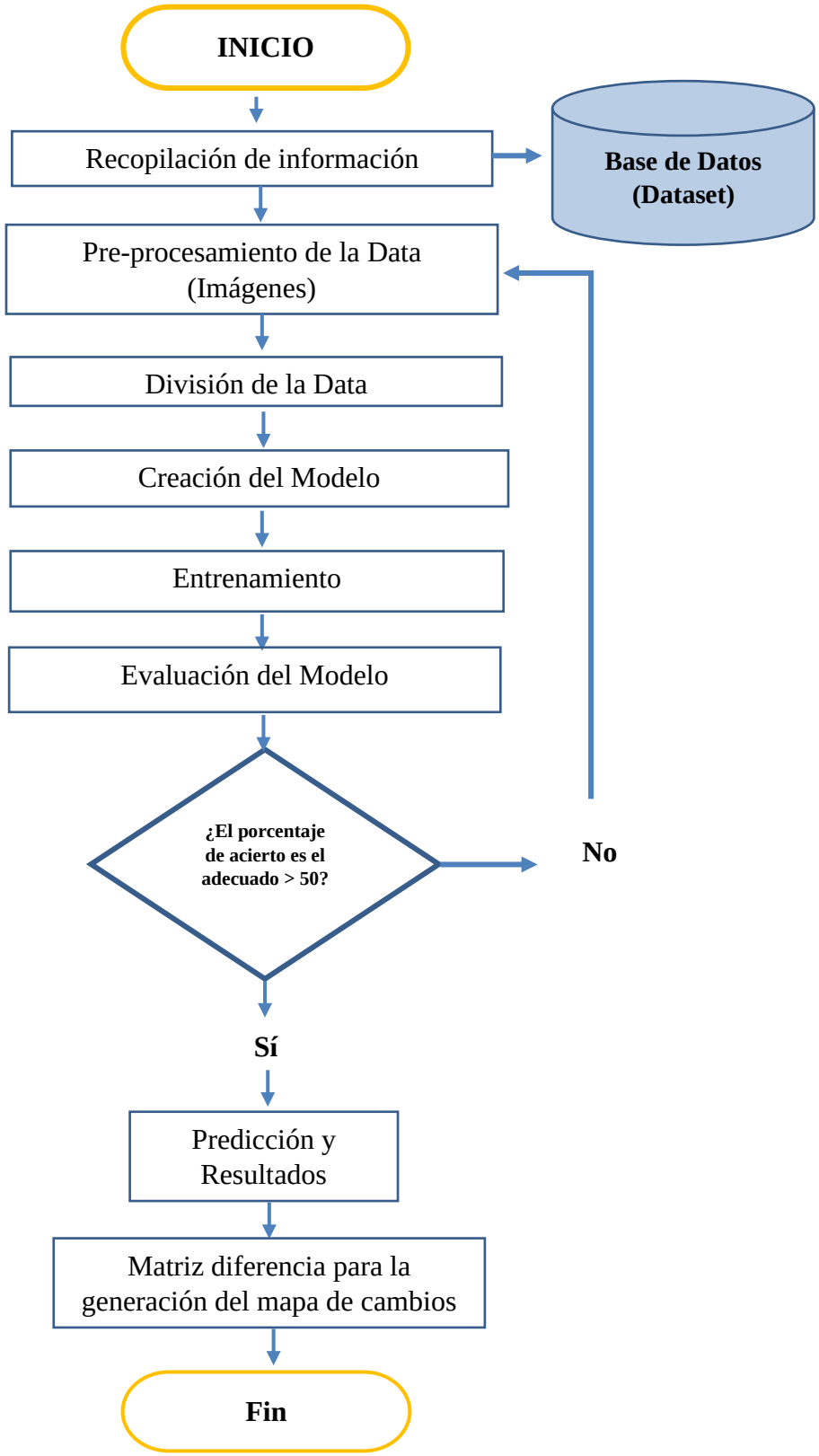

Figura 2. Diagrama de flujo de la metodología de trabajo.

5.1 Recopilación del Dataset

Las imágenes utilizadas para la generación de la base de datos se obtuvieron de la plataforma del Geoportal IGM, proporcionando 17 ortofotos de alta resolución en formato .tiff para el período de mayo de 2020 a mayo de 2021, que cubre la zona afectada por la erosión del Río Coca-Quijos (Figura 3).

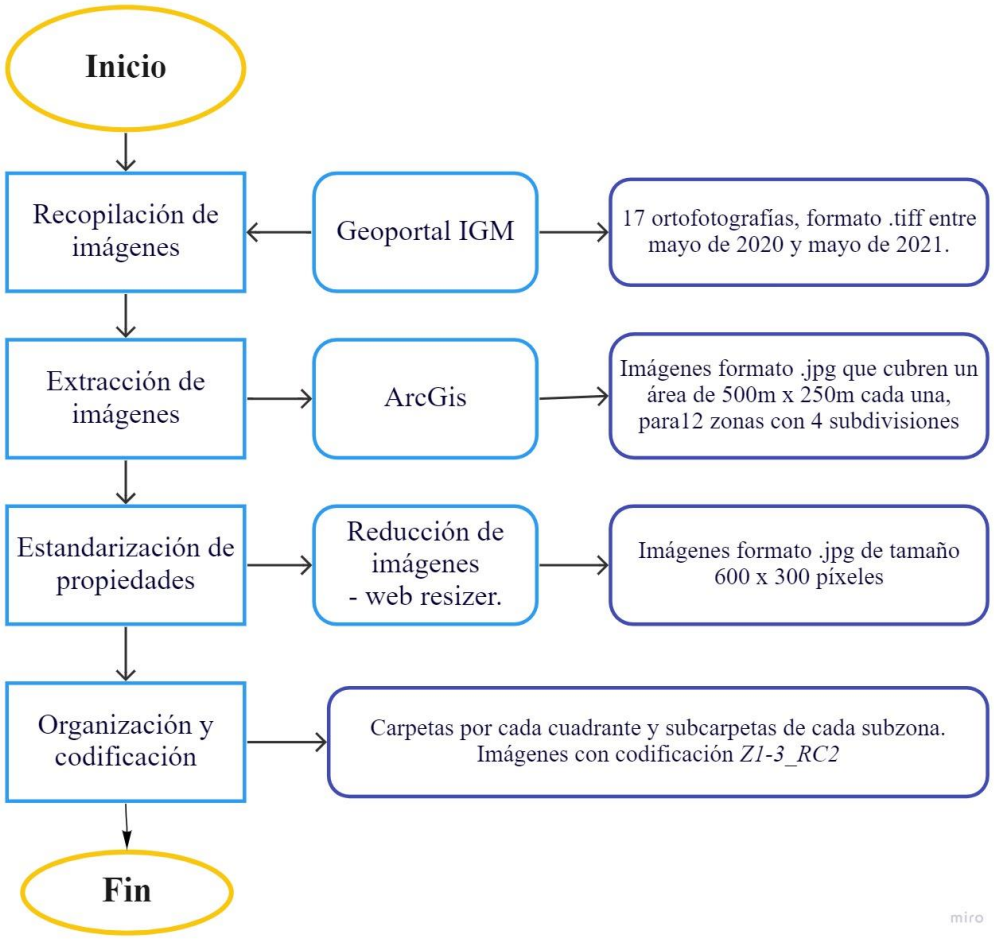

Figura 3. Diagrama de flujo de generación de la base de datos.

5.2 Pre-Procesamiento

El procesamiento de las ortofotografías se llevó a cabo en el software ArcGis que permitió recortar imágenes que cubren un área de 250m x 500m cada una, en formato .jpg. Posteriormente la reducción de las imágenes se realizó en una plataforma web (web resizer) generando imágenes con resolución de 300 x 600 pixeles. Este paso se lo realiza con la finalidad de optimizar el tiempo de procesamiento de información por píxel durante el posterior análisis en la red convolucional como muestra la figura 4.

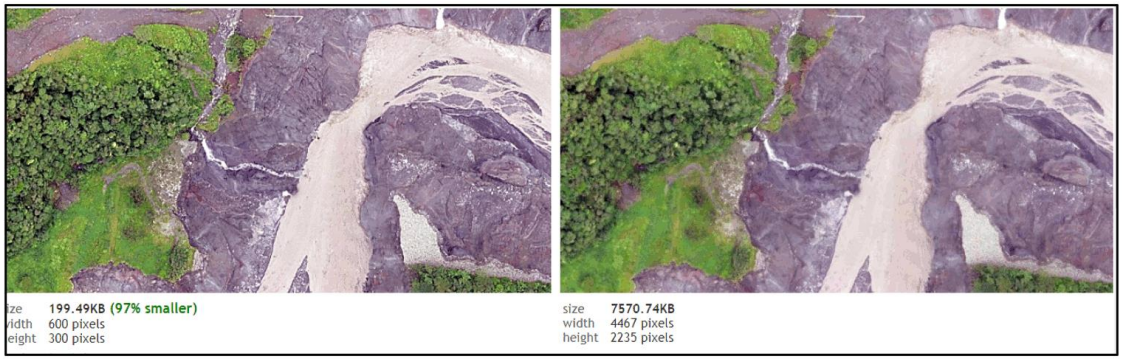

Figura 4.Reducción de la resolución de las imágenes

Para la organización del dataset se enumeraron los cuadrantes de 1 a 12 de norte a sur y cada uno se dividió en 4 subzonas (ver figura 5), además se agregó numeraciones de 1 a 17 para las ortofotografías en orden temporal siendo 1 la más antigua y 17 la más reciente como se evidencia en la tabla 2.

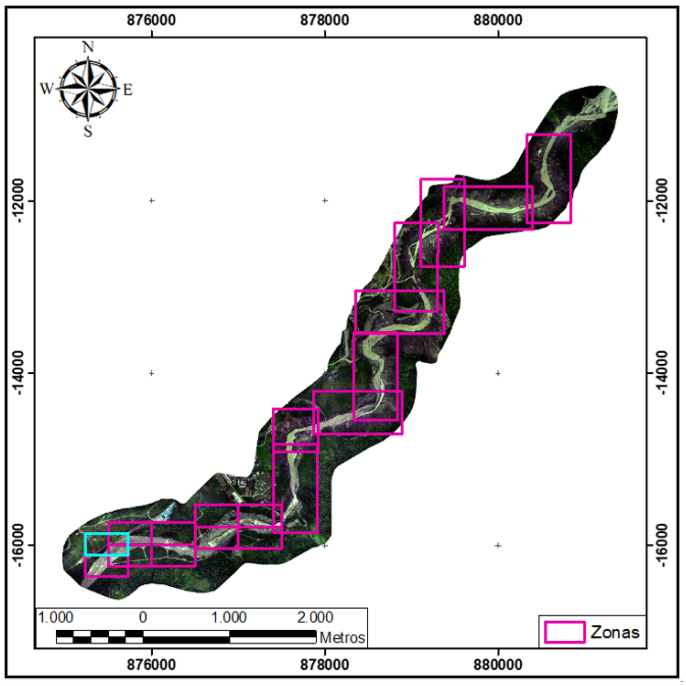

Figura 5.Cuadrantes establecidos para la zona de estudio correspondiente el Río Coca-Quijos

El dataset se organizó por carpetas de cada subzona, en cada carpeta consta el conjunto de imágenes correspondientes. Cada imagen fue codificada de la siguiente manera:

Z #zona - #subzona _ R #fotografía

Por ejemplo, la codificación Z10-1_R14 corresponde a la imagen de la época 14 de la zona 10, subzona 1.

Tabla 3. Codificación de las ortofotografías

Ortofotografías					
Código	**Fecha**	**Código**	**Fecha**	**Código**	**Fecha**
RC1	19/05/2020	RC7	29/07/2020	RC13	17/12/2020
RC2	10/06/2020	RC8	24/08/2020	RC14	15/01/2021
RC3	17/06/2020	RC9	02/10/2020	RC15	26/02/2021
RC4	01/07/2020	RC10	08/10/2020	RC16	21/04/2021
RC5	08/07/2020	RC11	13/11/2020	RC17	21/05/2021
RC6	22/07/2020	RC12	25/11/2020		

5.3 División de la Información

El dataset inicial comprende un total de 280 imágenes, de las cuales se selecciona el 25% para el test y el 75% para entrenamiento y validación (Tabla 4).

Tabla 4. División del dataset inicial.

	Porcentaje %	**N° de imágenes**
Entrenamiento y Validación	75	210
Evaluación	25	70
Total	100	**280**

Para el conjunto de entrenamiento y validación se realiza un balanceo de datos, el cuál es una técnica utilizada en machine learning para equilibrar todas las entradas que se procesaran en el modelo, con el proceso de balanceo se procura tener el mismo número de imágenes por cada una de las subzonas. El método utilizado para balanceo de datos corresponde al Random Over Sampler, el cual consiste en duplicar entradas en este caso imágenes representadas con un número minoritario. (Nafis, 2020). El resultado del balanceo dio un total de 42 imágenes por subzona.

Considerando que la data ha sido balanceada se procede a dividir un 75 % para entrenamiento y un 25% para validación, tomando 10 imágenes de cada categoría/subzona. La distribución de la data se puede observar en la tabla 5.

Tabla 5. División del conjunto de entrenamiento y validación.

	Porcentaje %	**Categorías**	**N° Imágenes**
Entrenamiento	75	17	170
Validación	25	6	60
Total	100	**23**	**230**

5.4 Creación del Modelo

Las redes siamesas son un tipo especial de arquitectura de redes neuronales. En lugar de que un modelo aprenda a clasificar sus entradas, las redes neuronales aprenden a diferenciar entre dos entradas y a resaltar la similitud entre ellos. (Alcaide, 2020). La CNN está compuesta por dos entradas es decir dos imágenes, dos capas de salida y varias capas intermedias ocultas, estas van desde las primeras capas simples hasta llegar a capas profundas. Estas capas realizan operaciones que alteran los datos con el objetivo de aprender características específicas de los mismos. (Alcaide, 2020).

Las entradas corresponden a un par de imágenes con un tamaño similar de pixeles, en este caso de estudio, ambas imágenes presentan una dimensión de 600 x 300 px. Las capas ocultas corresponden a capas convolucionales (CONV2D), las cuales van a requerir de un número de filtros que nos darán el número de mapas de características para cada capa convolucional. Por cada capa existen pequeños filtros representados por un KERNEL de 3X3, el cual tendrá la capacidad de analizar las imágenes y recolectar los rasgos más representativos pixel por pixel, de acuerdo a los pasos o STRIDES que estos realicen a través de toda la imagen y de un PADDING de (1,1), el cual brindará un mapa de características de igual tamaño, seguido de una función de activación ReLU (Figura 7).

Para ser más eficiente el proceso de abstracción y entrenamiento se aplica el MAXPOOLING que redimensionará las salidas en cada capa convolucional y extrae de los mapas de características la información más relevante, obteniendo nuevos mapas de características de acuerdo con el tamaño especificado del maxpooling para cada capa convolucional. (Figura 7)

En este proyecto se estableció 3 capas convolucionales con el fin de obtener características de bajo, medio y alto nivel. (Figura 6). Las salidas corresponderán a dos "encoder" que representan dos matrices con un conjunto de respuestas que son las representaciones o empaquetamiento de las características de la imagen obtenidas a través de la red convolucional. (Figura 7).

Posteriormente al tener nuestros dos encoder, se realiza una diferencia entre ambos mediante la función LAMBDA, dándonos como resultado una tercera matriz de igual dimensión que corresponde al resultado de esta operación, la cual se la denomino como: vector_result. (Figura 7).

Posteriormente el resultado de diferencia "vector_result", corresponde a la entrada de una red deconvolucional, la cual va a tener el objetivo de redimensionar esta matriz a un

tamaño similar que el de las dos imágenes ingresadas en el proceso convolucional (proceso inverso), para así obtener un “decoder” con dimensiones (300, 600, 3). En la última capa deconvolucional se aplica una función “Sigmoide”, la cual es una función matemática de activación que tendrá la propiedad de normalizar la respuesta de nuestro decoder de salida entre un rango de 0 y 1. (Figura 7).

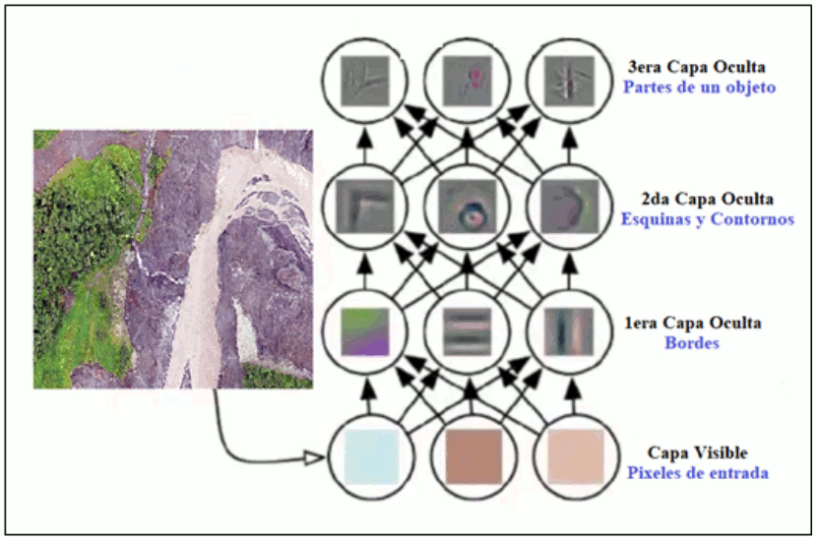

Figura 6. Proceso de extracción de características.

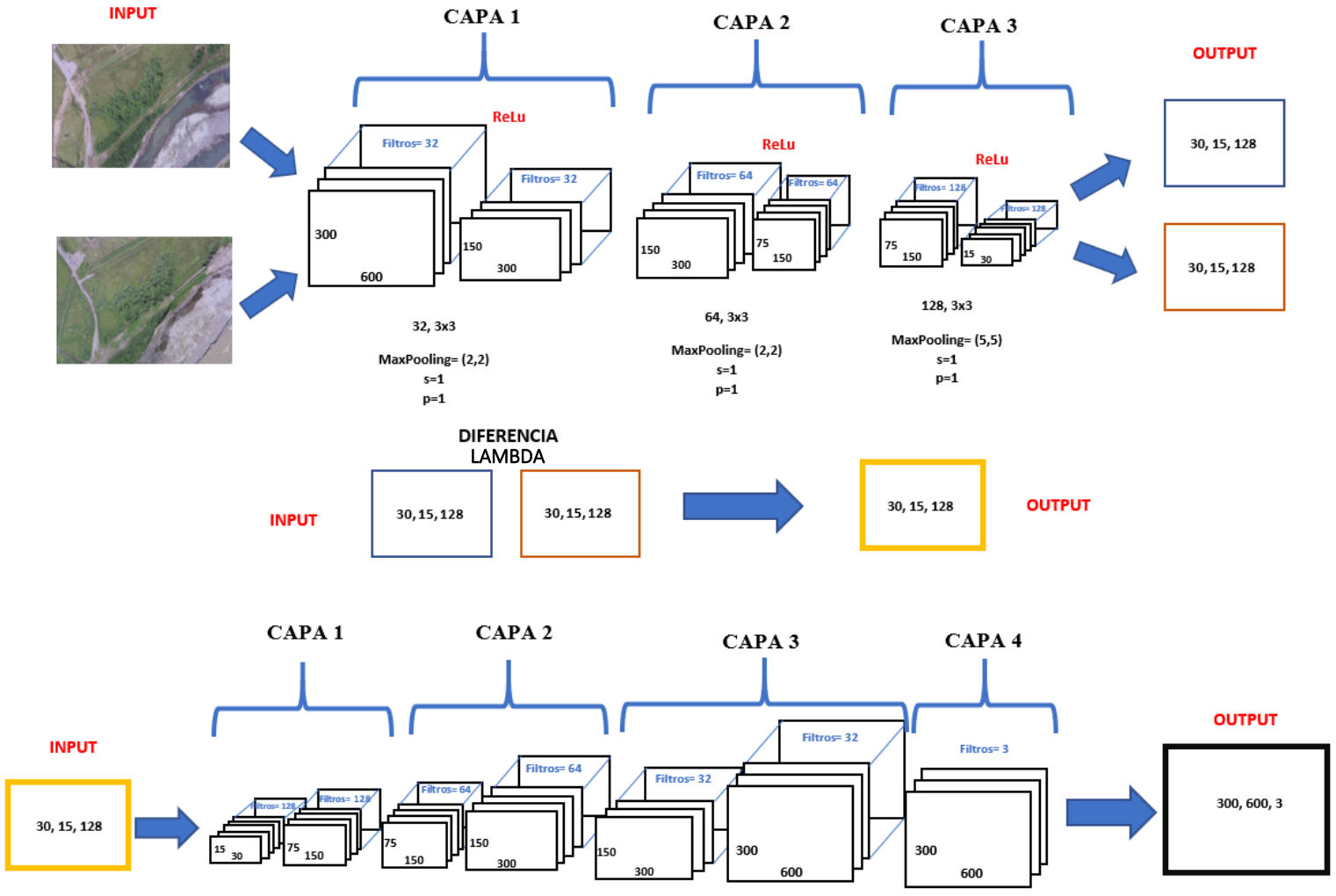

Figura 7. Arquitectura del modelo.

Los hiperparámetros utilizados para el modelo de la red convolucional se describen en la tabla 6.

Tabla 6. Hiperparámetros de la Red Convolucional.

CAPA 1	**Filtros:** 32, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** ReLU, **MaxPooling:** (2,2).
CAPA 2	**Filtros:** 64, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** ReLU, **MaxPooling:** (2,2).
CAPA 3	**Filtros:** 128, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** ReLU, **MaxPooling:** (5,5).

El número de parámetros fueron calculados mediante el producto de las dimensiones del Kernel (3x3), el número de filtros de cada una de las capas de la red y más el valor del Baias. (Figura 8).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 300, 600, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 150, 300, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 150, 300, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 75, 150, 64)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 150, 128)      73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 30, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 15, 30, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 57600)             0
_________________________________________________________________
dense (Dense)                (None, 128)               7372928
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 7,466,434
Trainable params: 7,466,434
Non-trainable params: 0
_________________________________________________________________
```

Figura 8. Sumario de Parámetros-Convolución.

Los hiperparámetros utilizados para el modelo de la red deconvolucional se describen en la tabla 7.

Tabla 7. Hiperparámetros de la Red Deconvolucional.

CAPA 1	**Filtros:** 128, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** ReLU, **UpSampling2D:** (5,5).
CAPA 2	**Filtros:** 64, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** ReLU, **UpSampling2D:** (2,2).
CAPA 3	**Filtros:** 32, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** ReLU, **UpSampling2D:** (2,2).
CAPA 4	**Filtros:** 32, **Kernel:** 3x3, **Padding:** (1,1), **Strides:** (1,1), **Función de activación:** SIGMOID.

Los parámetros de cada una de las capas en la red deconvolucional, se las calculo de manera similar como el proceso convolucional. (Figura 9).

```
1 Deco.summary()
```

```
Model: "DECODER"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
DeConv_1 (Conv2DTranspose)   (None, 30, 15, 128)       147584
_________________________________________________________________
UpSamp_1 (UpSampling2D)      (None, 150, 75, 128)      0
_________________________________________________________________
DeConv_2 (Conv2DTranspose)   (None, 150, 75, 64)       73792
_________________________________________________________________
UpSamp_2 (UpSampling2D)      (None, 300, 150, 64)      0
_________________________________________________________________
DeConv_3 (Conv2DTranspose)   (None, 300, 150, 32)      18464
_________________________________________________________________
UpSamp_3 (UpSampling2D)      (None, 600, 300, 32)      0
_________________________________________________________________
vector_Result (Conv2D)       (None, 600, 300, 3)       867
=================================================================
Total params: 240,707
Trainable params: 240,707
Non-trainable params: 0
```

Figura 9. Sumario de parámetros-Deconvolución.

5.5 Compilación y Ajuste

Se ingresa un par de imágenes a través de una Red Siamesa Convolucional para obtener pares de características representados en dos "encoder". A continuación, se utiliza una métrica de distancia predefinida simple o una función de diferencia, para medir la disimilitud de los pares de características o que tan cercanos a cero se da a partir de una resta de matrices para considerar su similitud. (Truong, 2020).

Posteriormente se tiene una matriz, en el cual los valores de diferencia son optimizados mediante el "`Optimizer Adam`". (Wang, 2019).

Debido a la escasa cantidad de información para poder entrenar el modelo de manera efectiva, en el caso de estudio se implementó un ONE SHOT, el cual posee la capacidad de generar modelos de aprendizaje profundo rápidos y precisos con menos muestras de entrenamiento. La función de entrenamiento ONE SHOT, ayudará a predecir de un solo disparo las características más notables de una entrada o par de entradas que se desea que aprenda el modelo. (Lamba, 2019).

La función utilizada para el aprendizaje es `nway_one_shot`, la cual es adecuada para el conjunto de datos utilizados ya que ayuda a implementar modelos de aprendizaje profundos, rápidos y precisos con una dataset limitada.

El aprendizaje del modelo utiliza los parámetros establecidos previamente en la estructura de la red siamesa, así como el conjunto de imágenes destinadas para entrenamiento y validación. Se han establecido lotes de entrenamiento de 64 imágenes seleccionadas aleatoriamente para cada época de aprendizaje a través de la función `get_batch` con su respectivo `batch_size=64`.

5.6 Evaluación

El entrenamiento se realizó con 10 hasta 100 épocas en donde el acierto del modelo fue medido a través del `accuracy` de cada época. Tanto para el conjunto de entrenamiento como para el conjunto de validación los valores del acierto no presentaron gran variación incluso al llegar a las 100 épocas, por lo que se tomó como tamaño ideal 10 épocas considerando que si se incrementaba este valor el acierto reducía. El accuracy llegó al 72,39% mientras que el `loss`, que representa la pérdida, presenta un valor de 50% y se

mantiene constante. Como se observa en la figura 10, se ve que la pérdida y el acierto no varían conforme se incrementan las épocas, esto se debe a la data set limitada con la que se está trabajando, que únicamente permite un leve incremento del accuracy.

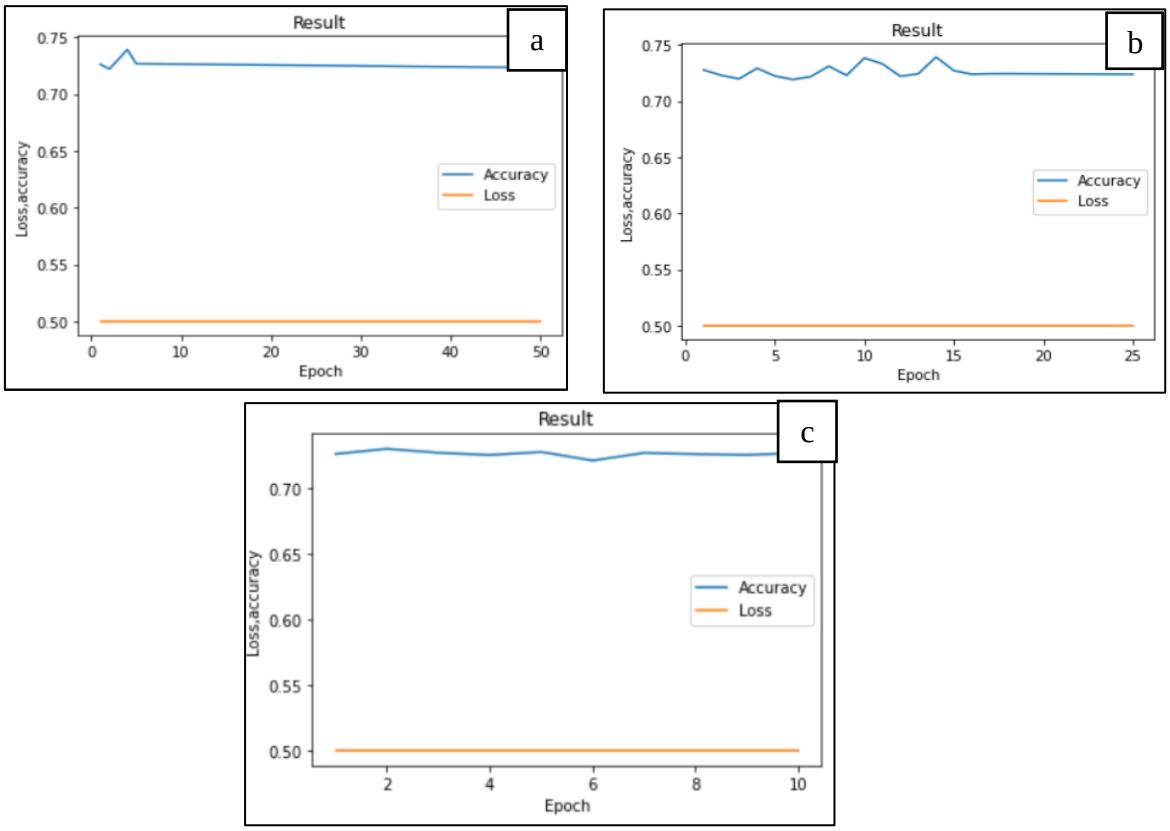

Figura 10. Resultados del accuracy y loss para a) 50 épocas, b) 25 épocas, c) 10 épocas.

5.7 Predicción

Para la etapa de predicción se utilizaron pares de imágenes correspondientes al conjunto de test, las cuales no fueron empleadas en el aprendizaje del modelo. Una vez ingresado el par de imágenes al modelo de la red se predice un encoder de diferencia denominado "`vector_result`", el cual mediante la función "`numpy array`" se logró convertirlo en una matriz denominada "`Matriz_result`" con valores comprendidos entre un rango de 0 y 1, donde los valores más cercanos a "cero" marcarán las zonas con alto grado de similitud, mientras que los valores más próximos a "uno" marcan las zonas donde los cambios son representativos, posteriormente a esta matriz se la transforma a un arreglo de 0 y 255 asemejándose a una imagen de 8bits, para después mediante el uso de "`plt.imshow`" se muestre una imagen que represente el mapa de cambios como se muestra en la figura 11. Donde las áreas que han cambiado se visualizan en color blanco,

mientras que las que no han sufrido cambio representativo se muestran en color negro como en la figura 11.

Figura 11. Se muestra el par de imágenes de un mismo sitio en dos tiempos diferentes y el mapa de cambios en blanco y negro.

6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

La base de datos más adecuada para ser procesada en una red neuronal convolucional debe tener los mismos parámetros para el correcto funcionamiento del modelo, por lo que las imágenes utilizadas se sometieron a procesos como recorte y reducción de su tamaño, además se notó que en el proceso de aprendizaje la data destinada debe estar balanceada para poder realizar un entrenamiento más eficiente.

El modelo convolucional implementado consta de 3 capas el cual ha facilitado la extracción de los parámetros necesarios para proceder a aplicar una función de diferencia, además de una red deconvolucional de 4 capas que permita redimensionar el resultado y posteriormente plotearlo como una imagen de 8 bits que identifique zonas donde se han presentado cambios. De esta manera se obtuvo una tripleta de imágenes en la que se observa un mismo escenario, pero cronológicamente distintos, y una tercera imagen que refleja los cambios morfológicos desarrollados por procesos de erosión.

La eficiencia del modelo se considera aceptable ya que, se obtuvo un valor de precisión superior a 0.7, pese a obtener un valor de perdida notable; el grado de certeza es admisible para la predicción ya que se obtuvieron buenos resultados que se corroboran visualmente en el mapa de cambios generado.

6.2 RECOMENDACIONES

Es recomendable que las imágenes pasen por una etapa de preprocesamiento más exhaustivo con el fin de reducir la influencia de la vegetación mediante otros programas alternativos, que permitan manipular la saturación, tono, brillo y contraste de las zonas donde se registran cambios debido a los efectos de sombra que se generan en estas áreas.

Para el proceso de entrenamiento es necesario que la data que se va a destinar se encuentre balanceada y tenga un número considerable de imágenes, con el fin de que el porcentaje de acierto tienda a aumentar y sea aceptable para la posterior predicción.

Se recomienda realizar un análisis exhaustivo de la parte bibliográfica previo a la elaboración del proyecto para una mejor comprensión de todas las partes que conforman la estructura metodológica del aprendizaje automático profundo, así mismo es de suma importancia tener una mayor preparación en el manejo del lenguaje Python.

7. ANEXOS

Código desarrollado para este proyecto:

https://github.com/p3pa24/Cambios_Rio_Coca/blob/main/Detecci%C3%B3n%20de%20cambios.ipynb

8. REFERENCIAS

Alcaide, A. (2020). “Redes Neuronales Convolucionales Siamesas aplicadas a la verificación facial". Colmanarejo: Creative Commons.

Buhigas, J. TensorFlow. Puentes Digitales. (14 de febrero de 2018). Recuperado de https://puentesdigitales.com/2018/02/14/todo-lo-que-necesitas-saber-sobre-tensorflow-la-plataforma-para-inteligencia-artificial-de-google/.

Buthi, P. (20 de junio de 2020). How To Build GIF/Video From Images With Python Pillow/Opencv. Recuperado de https://pomain.medium.com/how-to-build-gif-video-from-images-with-python-pillow-opencv-c3126ce89ca8.

Caballero, L. (2020). Programación en Python - Nivel básico. Covantec. Recuperado de 7.2. Manipulación de archivos — Materiales del entrenamiento de programación en Python - Nivel básico (entrenamiento-python-basico.readthedocs.io)

Fierro, A., Nakano, M., Yanai, K. y Pérez, H. (2019). “Redes convolucionales siamesas y tripletas para la recuperación de imágenes similares en contenido”. Información Yecnológica. 30 (6). 243-254.

Geoportal IGM. Instituto Geográfico Militar. “Repositorio: Ortofotografias del Rio Coca”. Recuperado: 01 de Julio del 2021, de http://www.geoportaligm.gob.ec/portal/.

Lamba, H. (2019). “Aprendizaje One Shot con redes siamesas usando Keras”. USA: Towards data science.

Matich, D. (2001). Redes Neuronales: Conceptos Básicos y Aplicaciones. Rosario-Argentina: Facultad Regional de Rosario.

Nafis, N. (2020). SacaTuCarrera. *“Balanceo de datos - #4 Machine Learning en Python”*. Recuperado de: https://www.youtube.com/watch?v=Ii1_NXobwKI.

Noemioc (21 de diciembre de 2020). Imageio Python. Github. Recuperado de https://noemioocc.github.io/posts/Crear-un-gif-con-imagenes-imageio-python/

Piperlab (2020). Keras. Business Data Science Differently. Recuperado de: https://piperlab.es/glosario-de-big-data/keras/

Salcedo, L. (11 de noviembre de 2017). Procesamiento de Imágenes con Python y Pillow. Mi diario Python. Recuperado de https://pythondiario.com/2017/11/procesamiento-de-imagenes-con-python-y-2.html.

Sanchez, A. (04 de octubre 2020). Manual de Python. La librería de Matplotlib. Wowchemy Website Builder. Recuperado de https://aprendeconalf.es/docencia/python/manual/matplotlib/

Sanchez, A. (04 de octubre 2020). Manual de Python. La librería de Numpy. Wowchemy Website Builder. Recuperado de https://aprendeconalf.es/docencia/python/manual/numpy/

Stocking, M., Murnaghan, N. (2003). "Manual para la evaluación de campo de la degradación de la tierra". Mundi Prensa Libros. España

Truong, L. (2020). "Detection of Road Surface Changes from Multi-Temporal Unmanned Aerial Vehicle Images Using a Convolutional Siamese Network". University of Mining and Geology: Department of Civil Engineering, Chonnam National University.

Wang, M. T. (2019). "A Deep Siamese Network with Hybrid Convolutional Feature Extraction Module for Change Detection Based on Multi-sensor Remote Sensing Images". Xuzhou-China: NASG Key Laboratory of Land Environment and Disaster Monitoring, China University of Mining and.

Suárez, A., Jiménez, A., Castro, M. & Cruz, A. (2017). "Clasificación y mapeo automático de coberturas del suelo en imágenes satelitales utilizando Redes Neuronales Convolucionales". Universidad de Los Llanos Meta. Orinoquia, Colombia. vol. 21.